
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV INFORMAČNÍCH SYSTÉMŮ**
**DEPARTMENT OF INFORMATION SYSTEMS**


# INTEGRACE CLOUDOVÝCH ÚLOŽIŠŤ DO WEBOVÝCH APLIKACÍ

**CLOUD STORAGE INTEGRATION INTO WEB APPLICATIONS**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** **ONDŘEJ STUDNIČKA**
**AUTHOR**

**VEDOUCÍ PRÁCE** **Ing. RADEK BURGET, Ph.D.**
**SUPERVISOR**



**BRNO 2021**

**Vysoké učení technické v Brně**
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

# Zadání bakalářské práce


Student: **Studnička Ondřej**
Program: Informační technologie
Název: **Integrace cloudových úložišť do webových aplikací**
**Cloud Storage Integration into Web Applications**
Kategorie: Web

Zadání:

1. Prostudujte současné technologie pro tvorbu webových aplikací s klientem v JavaScriptu.
2. Seznamte se s aplikačním rozhraním úložišť třetích stran jako např. Google Drive, One Drive, Dropbox, GitHub a dalšími a možnostmi jejich integrace do webové aplikace.
3. Po dohodě s vedoucím navrhněte architekturu webové aplikace pro interaktivní tvorbu prezentací uložených v cloudovém úložišti.
4. Implementujte navrženou aplikaci na vhodné platformě.
5. Proveďte testování vytvořené aplikace.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Gutmans, A., Rethans, D., Bakken, S.: Mistrovství v PHP 5, Computer Press, 2012
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Burget Radek, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 22. října 2020

# Abstrakt

Cílem práce je provést analýzu vybraných cloudových úložišť a jejich aplikačních rozhraní. Praktické využití je demonstrováno ve webové aplikaci, která slouží k tvorbě dynamických prezentací založených na frameworku Reveal.js. Aplikace umožňuje uživatelům ukládat vytvořené prezentace do zvoleného cloudového úložiště. Mezi podporovanými úložišti se nachází Google Drive, Dropbox, Github a Gitlab. Aplikace je implementována pomocí Vue.js. Teoretická část práce se zaměřuje na možnosti tvorby webových aplikací dle aktuálních trendů. Praktická část popisuje implementaci a testování vytvořené aplikace.



# Abstract

The aim of this thesis is to analyze selected cloud storages and their APIs. The practical use is demonstrated in web application that is used for to create dynamic presentation based on Reveal.js framework. The application allows user to save the created presentation to selected cloud storage. Supported storages are Google Drive, Dropbox, Github and Gitlab. The application is implemented using Vue.js. The theoretical part of the thesis focuses on the possibilities of creating web applications according to current trends. The practical part describes the implementation and testing of the created application.



# Klíčová slova

Javascript, Vue.js, Reveal.js, Google Drive, Dropbox, Github, Gitlab, API, webová aplikace, prezentace, cloudové úložiště

# Keywords

Javascript, Vue.js, Reveal.js, Google Drive, Dropbox, Github, Gitlab, API, web application, presentation, cloud storage


# Citace

STUDNIČKA, Ondřej. *Integrace cloudových úložišť do webových aplikací*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Burget, Ph.D.

# Integrace cloudových úložišť do webových aplikací

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Radka Burgeta, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Ondřej Studnička
3. května 2021


## Poděkování

Děkuji panu doktru Radku Burgetovi za vedení bakalářské práce, jeho ochotnou pomoc při konzultacích a poskytnuté rady a nápady. Dále bych chtěl poděkovat panu Martinu Macháčkovi za pomoc při tvorbě designu aplikace.

# Obsah

# Kapitola 1

# Úvod

S prezentacemi se člověk setkává téměř každý den. Jako podpůrný materiál jsou využívány učiteli při výuce, žáky při demonstraci řešení projektů, firmami při předkládání své práce zákazníkovi a určitě by se našlo mnoho další případů. V dnešní době, kdy svět zasáhla pandemie nemoci Covid-19, jejich význam ještě stoupá. Při vzdáleném prezentování se sdílenou obrazovkou ve značné míře odpadá lidský faktor, pomocí něhož by prezentující — za normálních okolností — publikum zaujal během živého vystoupení. Z toho důvodu je třeba hledat jiné cesty, jak vtáhnout posluchače do dění. Jednou z možností, jak tohoto cíle dosáhnout, je vytvoření poutavé a dynamické prezentace, která působí na vizuální smysly posluchače. S běžnými nástroji typu Microsoft PowerPoint však může být tvorba takových prezentací obtížná.

Tato bakalářská práce popisuje vývoj webové aplikace umožňující tvorbu dynamických prezentací uživatelům, kteří se alespoň základně orientují v jazycích HTML5 a CSS3. Pomocí těchto jazyků dokáže uživatel zobrazit v prezentaci téměř vše, na co si vzpomene. Je limitován pouze vlastní fantazií a zkušenostmi.

Ve druhé kapitole (2) je uveden výčet technologií pro tvorbu webových aplikací, cloudových úložišť a dostupných nástrojů pro tvorbu prezentací. Třetí kapitola (3) obsahuje informace o technologiích, které byly při vývoji použity, a cloudových úložištích, s nimiž aplikace komunikuje. Čtvrtá část (4) pojednává o návrhu samotné aplikace z pohledu grafického návrhu či její struktury. Kapitola pátá (5) se zabývá implementací. Zahrnuje informace o směrování, ukládání dat a jednotlivých funkcích aplikace. Šestá kapitola (6) obsahuje detaily k testování vytvořené aplikace. Poslední kapitolou (7) je pak závěr, který hodnotí provedou práci.

# Kapitola 2

# Stav současných technologií

Kapitola je členěna do tří částí. První (2.1) z nich vymezuje pojem webová aplikace a nabízí základní přehled jazyků a javascriptových frameworků užívaných k jejich tvorbě. Druhá část (2.2) se věnuje cloudovým úložištím a uvádí výčet těch nejpopulárnějších z nich. Poslední sekce obsahuje (2.3) informace o dostupných nástrojích pro tvorbu prezentací.

## 2.1 Webové aplikace

Jako webová aplikace se označuje aplikace taková, která je uživateli dostupná pomocí internetové sítě a webového prohlížeče. Aplikaci poskytuje uživateli webový server, a tak odpadá nutnost její lokální instalace. Lze k ní přistupovat z klasického počítače, tabletu, mobilního telefonu či podobného zařízení. Aplikace by měla být schopna správně pracovat bez ohledu na operační systém nebo webový prohlížeč [21].

Celosvětově nejpopulárnějším prohlížečem je Google Chrome[1], na druhém místě se nachází s velkým odstupem Safari[2], na třetím pak Firefox[3] [4]. Všechny tyto prohlížeče mají i svou mobilní aplikaci, tudíž mohou být používány na již zmíněných tabletech a mobilních telefonech.

Prohlížeč se stará o vyobrazení obsahu webu, který je dodán serverem, na obrazovce uživatele [7]. Ten může být buďto statický, nebo dynamický. V případě webových aplikací se jedná právě o obsah dynamický.

Dynamická stránka mění svůj obsah [14] na základě několika faktorů. Mezi základní faktory lze zařadit například čas — autor přidal na svůj blog nový příspěvěk — nebo interakci uživatele — přidání komentáře na sociální síti.

### 2.1.1 Jazyky

Existuje široká škála jazyků, které je možné využít k tvorbě webových stránek a úpravě jejich obsahu. Tato část popisuje základní z nich.

#### HTML

Základy jazyka HTML položil v roce 1991 Tim Berners-Lee [16]. Jedná se o jednoduchý značkovací jazyk, který definuje základní strukturu stránky [27]. Pomocí takzvaných tagů

[1] https://www.google.com/intl/cs_CZ/chrome/
[2] https://www.apple.com/safari/
[3] https://www.mozilla.org/cs/firefox/new/

dává význam různým blokům textu a umožňuje jejich vzájemné propojení. Nejaktuálnější verzí jazyka (v roce 2021) je verze 5.

#### CSS

První specifikace kaskádových stylů [32] vznikla v roce 1996. CSS se využívá pro formátování a vizuální modifikaci HTML. Elementům lze definovat nejrůznější vlastnosti — barva a velikost písma, pozadí, odsazení, obtékání a mnoho dalších. Nabízí možnost definice globálních stylů pro vícero elementů na základě společné vlastnosti — například atribut `class` v HTML. S nejaktuálnější třetí verzí má dokonce uživatel možnost vytvářet animace.

#### Javascript

Javascript [28] se objevuje v roce 1995. Jde o skriptovací jazyk, jež obohatil původně statické stránky o dynamické chování. Skript se odesílá do prohlížeče a teprve zde je zahájeno jeho vykonávání. Vkládá se přímo do HTML kódu. Jeho využití je široké. Může jít o validaci formulářových polí, manipulaci s obsahem stránky či nejrůznější výpočty — díky tomu dokáže i výrazně snížit zátěž serverů. Pomocí Node.js[4] jej lze využít právě i na serverové straně.

#### PHP

PHP [15] je velmi populární skriptovací jazyk. Na rozdíl od výše zmíněného Javascriptu pracuje výhradně na straně serveru. Prohlížeči se vrací pouze výsledek provedeného skriptu. Dokáže velmi dobře vytvářet dynamický obsah webu a jednoduše pracovat s databázemi. Dává uživatelům možnost vytvářet si šablony a následně je používat v různých částech webu. Staví na něm řada redakčních systémů pro správu webu.

### 2.1.2 Javascriptové frameworky

Javascriptové frameworky [11, 31] umožňují zrychlit a zefektivnit vývoj a zvýšit interaktivitu webu. Řeší za vývojáře rutinní úkoly, pomáhají s nekompabilitou prohlížečů, zvyšují přehlednost kódu. Rámce přináší možnost tvorby šablon, manipulace s DOM[5], přidání reakcí na události vyvolané uživatelem či snadnější tvorbu pokročilých prvků. Mezi populární řešení patří React[6], Angular[7] nebo Vue.js[8], který v sobě kombinuje nejlepší vlastnosti předešle zmíněných.

## 2.2 Cloudová úložiště

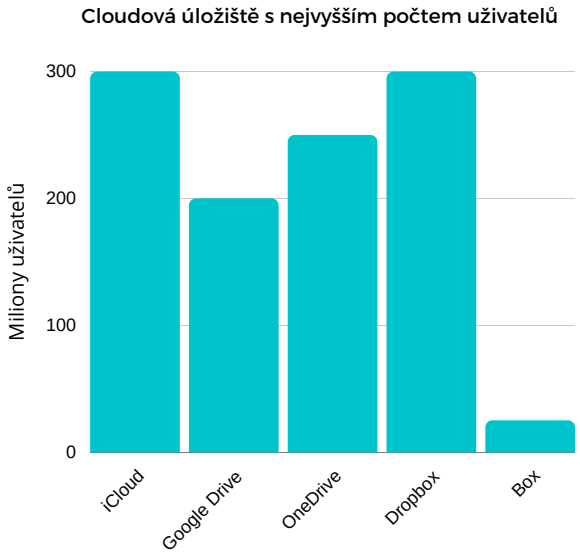
Pojmem cloud [5] se označuje síť propojených vzdálených serverů, k nimž uživatelé přistupují pomocí internetové sítě. Rozlišují se čtyři základní typy, a sice veřejný, privátní, hybridní a komunitní. Cloud může být použit k takzvanému cloud computingu[9], z pohledu této práce je však významnější možnost využít jej jako úložiště.

[4] https://nodejs.org/en/
[5] DOM – objektově orientovaná reprezentace XML nebo HTML dokumentu [25]
[6] https://reactjs.org/
[7] https://angular.io/
[8] https://vuejs.org/
[9] Cloud computing – doručování výpočetních služeb přes internet

Cloudové úložiště [6] je služba, která uživateli umožňuje ukládat svá data na vzdáleném úložném systému třetích stran. Využití úložišť je poměrně široké — ukládání médií, zálohování emailů, kontaktů, SMS zpráv či firemních dat, dříve také hostování webových stránek.

### 2.2.1 Známá cloudová úložiště

Mezi cloudová úložiště s největším počtem uživatelů se řadí iCloud[10], Dropbox[11] či Google Drive[12] [26]. S těmito úložišti pracují stovky milionů lidí po celém světě. Kromě přímého přístupu pomocí webového prohlížeče nabízejí i možnost stažení aplikace do počítače. V základu nabízejí omezený počet prostoru pro soubory zdarma, další lze dokoupit za jednotky dolarů[13]. Své uživatele si jistě najdou i ta méně známá – pCloud, Mega, MediaFire, Sync.com, Yandex Disk... Jako netradiční cloudové úložiště mohou být označeny i služby typu Github[14]. Jedná se o webové služby, které podporují vývoj software pomocí nástroje Git[15].




Obrázek 2.1: Cloudová úložiště s nejvyšším počtem uživatelů [26]

### 2.2.2 Aplikační rozhraní cloudových úložišť

Cloudová úložiště mohou nabízet možnost integrace do aplikací třetích stran. Příklad takové integrace představuje služba Draw.io[16], která uživatelům dává možnost vytvořit si diagram

[10] https://www.icloud.com/
[11] https://www.dropbox.com/
[12] https://www.google.com/intl/cs_CZ/drive/
[13] https://www.businessinsider.com/best-cloud-storage-price-google-drive-dropbox-icloud-one-drive-2014-12
[14] https://github.com/
[15] Git – Verzovací systém
[16] https://app.diagrams.net/

a ten následně uložit do svého cloudového úložiště. Aplikace s úložišti komunikuje pomocí aplikačního rozhraní (API) — soubor procedur, funkcí, protokolů a knihoven využívaný v rámci vývoje software [17]. Většina úložišť používá typ REST [10], který se orientuje na data, je bezstavový a využívá HTTP protokol. Komunikace probíhá pomocí vytváření HTTP požadavků na koncové body, ty mají formát URL adresy. Jsou využívány metody `GET`, `POST`, `PUT` a `DELETE` (případně `PATCH`). Pomocí těchto požadavků se dají v úložišti vytvářet, mazat či aktualizovat soubory, získávat informace o uživatelích... Na každý požadavek je vrácena patřičná odpověď, v jejíž hlavičce lze nalézt stavový kód. Kód upřesňuje jaká byla reakce serveru na klientský požadavek.

### 2.2.3 Autorizace OAuth 2.0

Aby mohla aplikace s úložištěm komunikovat, musí jí uživatel povolit přístup ke svému účtu. K tomuto účelu se hojně využívá autorizace OAuth 2.0 [20]. Pro zprovoznění autorizace uživatelům musí vývojář aplikace podniknout několik kroků, jež jsou společné napříč službami.

V první řadě si musí u dané platformy vytvořit svůj účet. Po jeho vytvoření nalézt sekci „*Pro vývojáře*“ nebo podobně. Zde je třeba zaregistrovat nově vyvíjenou aplikaci. Průběh registrace a množství požadovaných informací se v jednotlivých službách liší. Nejdůležitějším bodem pro následné programování je správné nastavení návratové URL adresy, kam bude uživatel po autorizaci přesměrován. Po dokončení registrace se aplikaci přiděluje dvojice údajů ClientID a ClientSecret. ClientID označuje unikátní ID aplikace v rámci dané platformy, na ClientSecret by se dalo zjednodušeně nahlížet jako na heslo — hodnotu zná pouze aplikace a autorizační server. Právě autorizačnímu serveru se aplikace těmito údaji prokazuje.

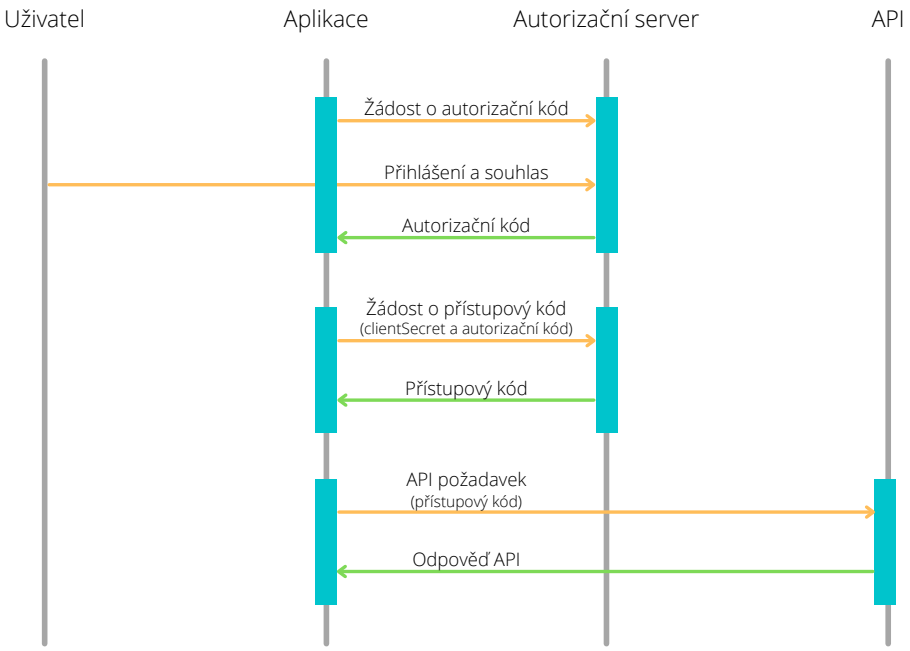
Samotná autorizace probíhá v následovném pořadí. Uživatel vyvolá požadavek na přihlášení a je přesměrován na stránku organizace. URL pro přesměrování je doplněna několika parametry:

- `client_id` – vygenerované ID aplikace,
- `redirect_uri` – adresa pro přesměrování po povolení/zamítnutí přístupu,
- `scope` – úroveň oprávnění.

U některých služeb lze přidat další parametry v závislosti na požadovaném chování:

- `state` – náhodný řetězec,
- `response_type` – typ odpovědi,
- a další.

Zde uživatel může aplikaci přístup ke svému účtu povolit nebo zamítnout. V obou případech je přesměrován zpět do aplikace na zadanou `redirect_uri`, která je doplněna parametrem o výsledek uživatelova rozhodnutí. V případě, že uživatel aplikaci povolil přístup ke svému účtu, se v URL nachází parametr `code`. Aplikace přečte hodnotu tohoto parametru a vytvoří nový požadavek, v němž hodnotu společně s ClientSecret zašle autorizačnímu serveru. Aplikaci je v odpovědi vrácen `access_token`. Tento token slouží k prokazování při volání API.




Obrázek 2.2: Tok událostí během autorizace a následné volání API [29]

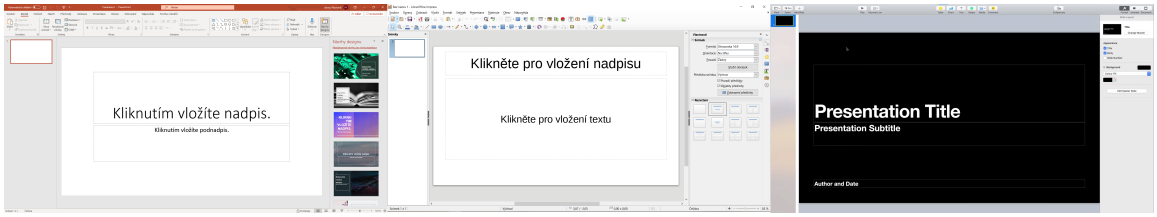
## 2.3 Nástroje pro tvorbu prezentací

Prezentace je dnes možné tvořit lokálně u sebe na počítači i pomocí webových aplikací. Nástrojů k tomu určených má uživatel k dispozici celou řadu. Níže jsou zmíněny některé z nich.

### 2.3.1 Desktopové aplikace

Desktopové aplikace nabízejí svým uživatelům širokou škálu šablon, všemožných nastavení, animací, přechodů... Některé jsou nabízeny zdarma, jiné zase za poplatek. V obou případech však často bývají dodávány jako součást kompletního kancelářského balíčku. Mezi nejznámější aplikace patří Microsoft PowerPoint[17], Impress[18] spadající pod LibreOffice či Keynote[19] pro majitele zařízení od firmy Apple Inc. Zdatnější uživatelé mohou sáhnout po nástroji LaTeX[20] rozšířeným o balíček Beamer, s jehož pomocí lze prezentace také vytvářet. S rozvojem technologií však i tyto zmíněné nástroje dostaly podobu webových aplikací, lze je tak využívat oběma způsoby.

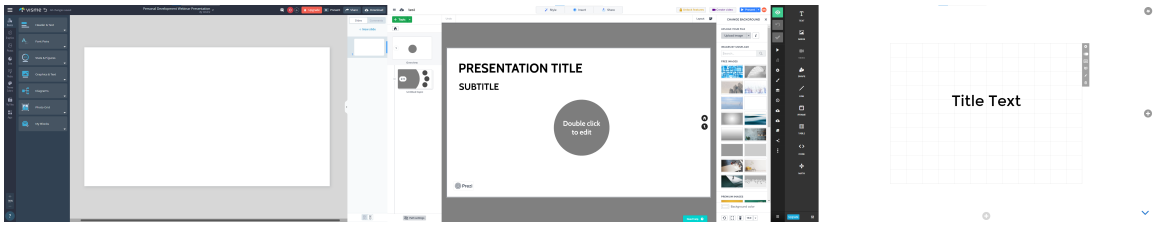
[17] https://www.microsoft.com/cs-cz/microsoft-365/powerpoint
[18] https://cs.libreoffice.org/discover/impress/
[19] https://www.apple.com/keynote/
[20] https://www.latex-project.org/




Obrázek 2.3: Uživatelská prostředí aplikací PowerPoint, Impress a Keynote

### 2.3.2 Webové aplikace

Webové aplikace oproti těm desktopovým nemusí disponovat tak širokou paletou nástrojů a možností. I přes to se však jedná o plnohodnotné nástroje. Kromě tvorby prezentací často nabízí i další možnosti využití. Příkladem může být tvorba infografiky, bannerů pro sociální sítě či úprava videa. Uživatel si může zvolit z několika úrovní cenových plánů, přičemž platí pravidlo — čím vyšší cena, tím více možností. Rozšíření se může týkat počtu dostupných šablon, prezentací, jež uživatel může spravovat pod jedním účtem, velikosti prostoru pro ukládání prezentací, možnosti stahovat v různých formátech a dalších podobných vlastností. Populárními aplikacemi jsou Visme[21], Prezi[22], Slides[23] nebo Google Slides[24], které spolupracují s Google Drive.




Obrázek 2.4: Uživatelská prostředí aplikací Visme, Prezi a Slides

[21] https://www.visme.co/
[22] https://prezi.com/
[23] https://slides.com/
[24] https://www.google.com/slides/about/

# Kapitola 3

# Použité technologie a zvolená cloudová úložiště

Kapitola se člení do tří částí, které obsahují informace o technologiích a úložištích, jež jsou v rámci aplikace využívány. První z nich (3.1) pojednává o frameworku Vue.js a jeho možnostech. Následně navazuje sekce 3.2 o správci balíčků NPM a použitých balíčcích. Zvláštní pozornost je věnována balíčku Reveal.js, jehož úlohou je tvorba dynamických prezentací. Kapitolu uzavírá pasáž 3.3 o zvolených cloudových úložištích.

## 3.1 Vue.js

Vue.js[1] [3, 18, 19] je progresivním javascriptovým rámcem, jenž se v současné době hojně využívá k tvorbě reaktivních[2] aplikací. Může se jednat o aplikace webové i mobilní (v takovém případě se využívá Vue Native[3]). Za frameworkem stojí Evan You, při vývoji se inspiroval nejlepšími vlastnostmi rámců React a Angular.

Vue.js se těší vysoké popularitě hned z několika důvodů. Prvním z nich je malá datová velikost, díky čemuž není nijak výrazně ovlivněna rychlost načítání webu. Tvůrci uvádí, že framework lze velmi jednoduše implementovat i do již existujícího řešení. Vývoj jednotlivých částí aplikace může probíhat nezávisle na sobě, což celý proces může značně zrychlit. Syntaxe od sebe odděluje HTML, CSS a Javscript, díky tomu je vše přehledné a snazší na pochopení — strmá učící křivka. Opomenuta nesmí být ani kvalitní dokumentace a široká komunita okolo.

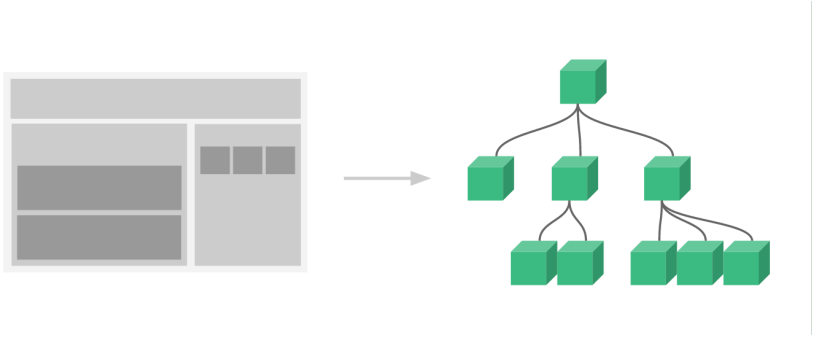
Mezi základní prvky Vue.js se řadí virtuální DOM, který pomáhá ušetřit výpočetní sílu, a komponenty, jimž se věnuje následující podkapitola. Na virtuální DOM [8] by se dalo pohlížet jako na virtuální kopii HTML DOMu. V momentě, kdy proběhne nějaká změna v DOM, se porovná požadovaný DOM (v této fázi virtuální) s původním modelem. Hledají se změny mezi oběma modely a následně probíhá aktualizace. Aktualizují se jen skutečně změněné prvky.

### 3.1.1 Komponenty

Systém komponent [2, 36] lze označit jako nejdůležitější prvek celého frameworku. Každá komponenta v podstatě odpovídá jedné instanci Vue. Nese si vlastní šablonu, data i logiku.

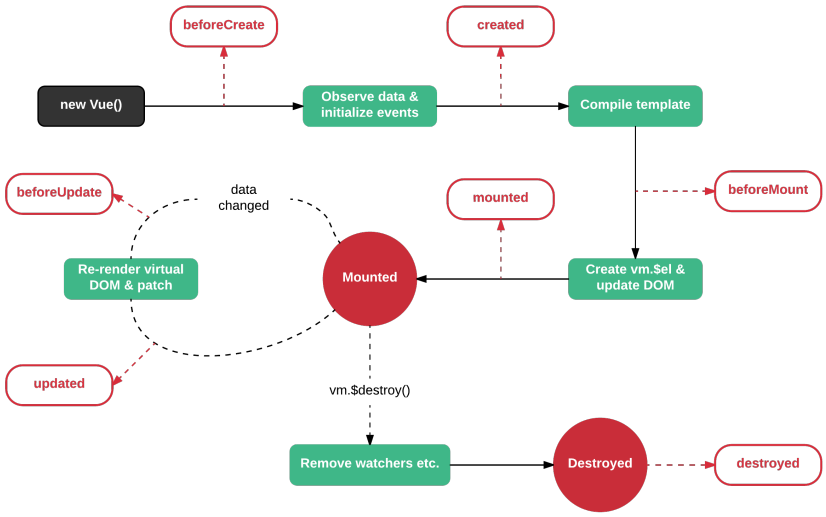
[1] https://vuejs.org
[2] Reaktivita – schopnost zobrazení reagovat na změnu stavu aplikace [13]
[3] https://vue-native.io/

Komponenty lze využívat opakovaně a nezávisle na sobě bez ovlivňování. Komponenta může obsahovat další komponenty (ty je třeba registrovat), díky čemuž vzniká hierarchie.



Obrázek 3.1: Ukázka možné hierarchie komponent [36]

Každá komponenta má svůj životní cyklus. Jedná se o tok událostí, kterými prochází, a vývojář má možnost na každou z nich reagovat. Události lze doplňovat o `nextTick`, díky čemuž lze provádět například akce po tom, co se změnila data, ale prohlížeč ještě nestihl na změnu zareagovat. Stěžejními body v životním cyklu jsou stavy `created` (komponenta již obsahuje data) a `mounted` (sestavená šablona je již viditelně vložena).




Obrázek 3.2: Životní cyklus komponenty [1]

### Šablony

Syntaxe šablony je založena na jazyku HTML. To je rozšířeno o takzvané v-direktivy (`v-if`, `v-for`, ...), na které lze pohlížet jako na obyčejné funkce v PHP. Vue.js však přidává i vlastní direktivy – `v-on` pro odchycení události (kliknutí, zmáčknutí klávesy, ...), `v-show` pro podmíněné zobrazení a další. Rámec umožňuje vývojářům vytvářet také vlastní direktivy. Pokud chce vývojář vypsat hodnotu proměnné, pak použije takzvaný *Mustache* zápis – `{{message}}`. Tato syntaxe však nelze použít pro vytvoření HTML atributů, zde musí být použita direktiva `v-bind`.

### Data, props

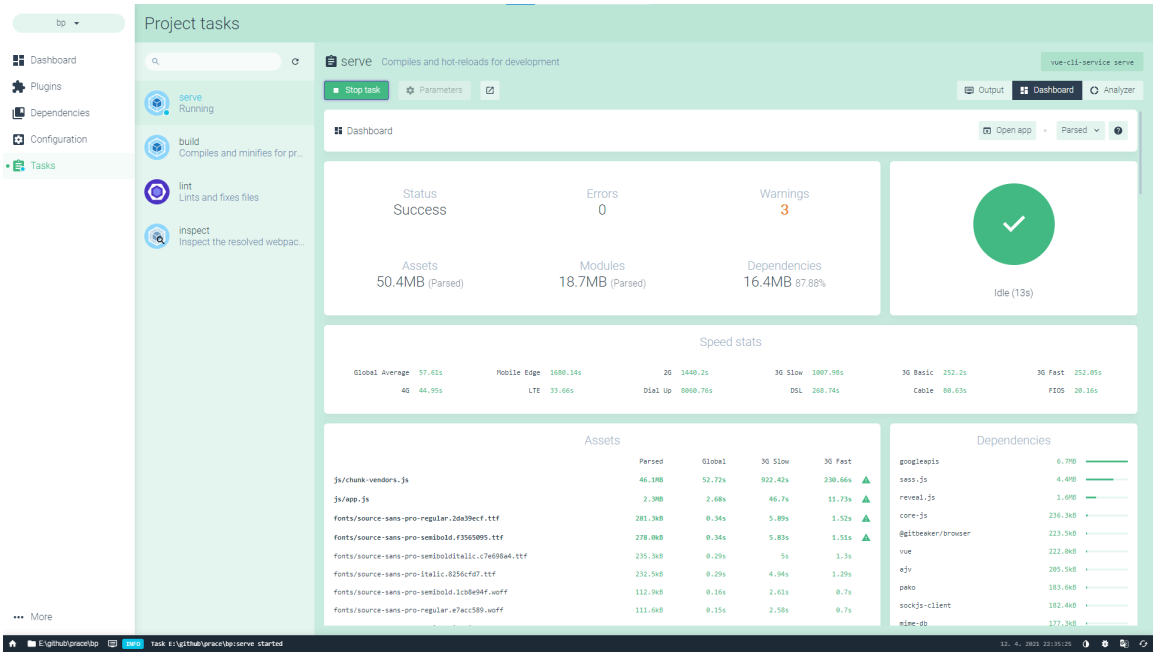
Data s nimiž komponenta pracuje jsou uchovávána ve funkci [35], která vrací objekt s položkami, k nimž je možné následně přistupovat. Položka odpovídá proměnné. Položky mohou být modifikovány i jinými komponentami. Pokud komponenta potřebuje využívat data rodičovské komponenty, pak může být vhodné tato data předávat pomocí takzvaných `props` [30]. Předání probíhá opět pomocí direktivy `v-bind`.

### Methods, computed, watchers

Jedná se o objekty metod, ve kterých může vývojář definovat své vlastní funkce, modifikovat data či sledovat změny dat. Objekt `methods` slouží k definici funkcí. Vhodným příkladem využití `computed` může být změna formátu data z `YYYY-MM-DD` na `DD. YY. YYYY`. `Watch` může být užitečné v případě, kdy aplikace musí výrazněji zareagovat na změnu dat. Při změně lze porovnat novou a původní hodnotu (pouze pro jednoduché datové typy) či zavolat funkci z `methods`. Sledovat změny hodnot objektu je možné pomocí parametru `deep`. Každá metoda pro sledování změny stavu je pojmenována stejně jako sledovaná proměnná.

## 3.1.2 Vue CLI

Jedná se o soubor základních nástrojů [23, 34] pro práci s ekosystémem Vue. Jeho využití je výhodné v případě, že celá aplikace je vyvíjena pomocí Vue.js. Pomáhá se sestavením aplikace, konfigurací knihoven a dalšími nastaveními, díky čemuž se mohou vývojáři soustředit převážně na vlastní kódování. Mimo jiné nabízí jednoduchý nástroj Vue UI, díky kterému se vývojáři mohou přesunout od používání příkazové řádky k přehlednému grafickému rozhraní.




Obrázek 3.3: Prostředí Vue UI

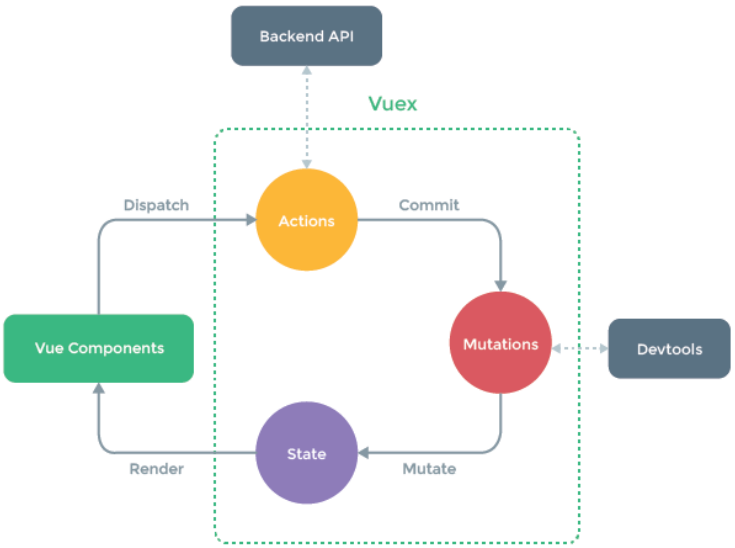
### 3.1.3 Vue router

Vue router [9] je oficiálním směrovačem aplikací Vue. Směrovač pomáhá s mapováním komponent na jednotlivé URL adresy, přidáváním dynamických parametrů do URL či přesměrováním. Ve výchozím stavu si router vypomáhá přidáním *hashe* do adresy. Toto chování lze potlačit pomocí nastavení možnosti `mode` na hodnotu `history`.

### 3.1.4 Vuex

Vuex [9, 33] zprostředkovává správu stavů aplikace, jež jsou dostupné z každé komponenty. Zjednodušeně řečeno se jedná o sklad globálních proměnných, díky kterému se výrazně zjednodušuje předávání dat napříč aplikací. Základem každého takového skladu jsou objekty:

- `state` – pro uchovávání dat,
- `mutations` – funkce ke změně stavu,
- `actions` – funkce k volání mutací,
- `getters` – funkce ke čtení dat ze `state`.

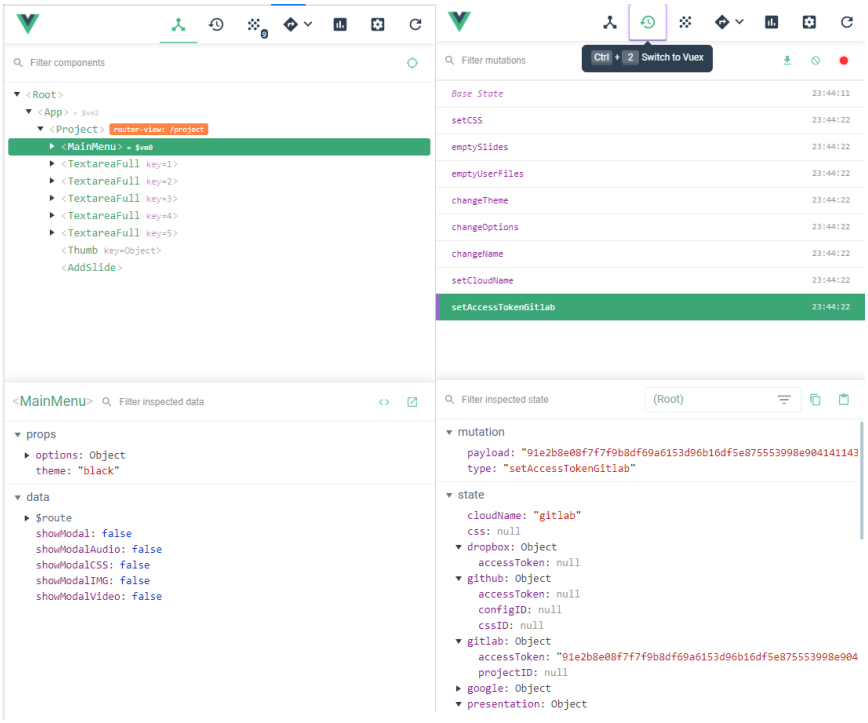



Obrázek 3.4: Schéma Vuex [33]

### 3.1.5 Vue.js devtools

Vue.js devtools[4] je rozšířením prohlížečů, které usnadňuje ladění vyvíjené aplikace. Nástroj dokáže automaticky rozpoznat webové aplikace vytvořené pomocí Vue.js. Lze zkoumat strukturu aplikace, události, data, ale také výkonnost.

[4] https://github.com/vuejs/vue-devtools




Obrázek 3.5: Prostředí Vue.js devtools v prohlížeči Chrome

## 3.2 Node package manager

Node package manager[5] (zkráceně npm) [22, 24] je největším registrem volně dostupného softwaru na světě. Nástroj umožňuje instalovat a spravovat balíčky za účelem rozšíření aplikace nebo sdílet vlastní kód ostatním uživatelům. Obvykle se ovládá pomocí příkazové řádky, Vue UI však nabízí možnost jeho použití v prostředí prohlížeče.

Instalace balíčku se provádí pomocí příkazu `npm install package` (pomocí přidání `@x` za název balíčku lze specifikovat požadovanou verzi). Moduly se ukládají do složky `node_modules`. Informace o všech použitých balíčcích jsou obsaženy v rámci souboru `package.json`. Díky tomu si vývojáři nemusí mezi sebou předávat všechny zdrojové kódy. Stačí předat tento soubor a druhá strana si požadované závislosti sama doinstaluje.

### 3.2.1 Reveal.js

Za nejdůležitější z použitých modulů lze označit Reveal.js [12]. Jedná se o framework založený na webových technologiích sloužící k tvorbě dynamických prezentací. Za jeho vývojem stojí Hakim El Hattab. Pro práci s Reveal.js musí uživatel mít alespoň elementární znalosti HTML nebo Markdown, případně CSS.

Základní myšlenkou rámce je, že tag `section` představuje jeden snímek prezentace. Snímky lze do sebe zanořovat. Využití této možnosti je vhodné k přehlednému strukturování prezentace. Tag `section` může být doplněn o různé atributy, kterými se ovládá vzhled či chování prezentace. Příkladem může být nastavení barvy pozadí snímku pomoci atributu `data-background-color`.

[5] https://www.npmjs.com/

V základu obsahuje Reveal.js jedenáct šablon a šest pluginů. Uživatel si však může jednoduše vytvářet vlastní. Pluginy umožňují následující:

- RevealHighlight – prezentování zdrojového kódu,
- RevealMarkdown – používání Markdown zápisu ve snímcích,
- RevealSearch – vyhledávání v prezentaci,
- RevealNotes – poznámky pro prezentujícího,
- RevealMath – sázení matematických symbolů,
- RevealZoom – přiblížení detailu snímku.

Kromě kódu a matematických symbolů jdou na snímky umisťovat také obrázky a videa. Ty mohou být použity i jako pozadí snímku. Prezentaci lze inicializovat s mnoha nastaveními[6]. Dokončená prezentace se dá exportovat do formátu PDF.

### 3.2.2 Další použité balíčky

V tabulce se nachází výčet balíčků, které jsou v rámci aplikace využívány. U každého z nich je uveden název a účel jeho použití.

Tabulka 3.1: Další použité balíčky v rámci aplikace

| **Název balíčku** | **Využití** |
|---|---|
| @gitbeaker/browser | Komunikace s aplikačním rozhraním služby GitLab |
| @octokit/core | Komunikace s aplikačním rozhraním služby GitHub |
| csstree-validator | Validace uživatelem vloženého CSS |
| gdrive-utils | Komunikace s aplikačním rozhraním služby Google Drive |
| latinize | Převod národních znaků |
| prismjs | Zvýrazňování syntaxe ve formulářových polích |
| dropbox | Komunikace s aplikačním rozhraním služby Dropbox |
| sass.js | Převod SCSS do CSS |
| vodal | Modální okno |
| vue-file-agent | Kontrola a nahrávání souborů pomocí Drag&Drop |
| vue-prism-editor | Transformace formulářového pole do editoru kódu |

## 3.3 Zvolená cloudová úložiště

Aplikace komunikuje s čtyřmi cloudovými úložišti, a sice Google Drive, Dropbox, GitHub a Gitlab. Níže jsou uvedeny bližší informace ke každému z nich. Všechna úložiště využívají autorizaci pomocí protokolu OAuth, která byla popsána v kapitole 2.2.3. Úložiště mohou poskytovat vývojářům souhrnné statistiky o využívání aplikací.

[6] https://github.com/hakimel/reveal.js/blob/master/js/config.js

### 3.3.1 Google Drive

Google Drive je jedním z nejpoužívanějších úložišť na světě. Uživatelům nabízí zdarma 15 GB prostoru pro jejich data.

Registrace nové aplikace se provádí ve službě Google Cloud Platform[7], která je společná pro všechny poskytované produkty společnosti Google. Zde je třeba založit nový projekt a vyplnit požadované informace. Aplikace dostane přiděleno dříve zmiňované ClientID a ClientSecret (souhrnně Credentials). Posledním krokem je aktivace požadovaného API — v tomto případě Google Drive API[8].

Google Drive je zajímavý hned z několika pohledů, a sice umožňuje vytvářet soubory a složky se stejným názvem a stejnou cestou, soubory a složky se rozlišují pouze pomocí typu MIME[9]. Vývojáři také musí pamatovat na fakt, že pro nahrání souboru do složky musí znát ID složky, nestačí zadat pouze cestu.

### 3.3.2 Dropbox

Také Dropbox se těší vysoké popularitě uživatelů. Základní plán je oproti Google Drive o poznání chudší, uživatelé mají k dispozici pouze 2 GB prostoru.

Novou aplikaci registruje vývojář pod svým účtem ve vývojářské sekci[10]. Během registrace se specifikuje, zda aplikace potřebuje plný přístup k datům uživatele nebo jen přístup ke své složce. Tato složka je následně vytvořena každému uživateli, jež se rozhodne danou aplikaci využívat.

Dropbox nabízí vývojářům předpřipravené nástroje pro komunikaci se svým aplikačním rozhraním v různých jazycích doplněné o ukázky použití (pro jazyk Javascript[11]). V obsáhlé dokumentaci[12] lze najít všechny potřebné informace. Dropbox — jako jediné ze zvolených úložišť — dává vývojářům možnost nahrávat více souborů pomocí jedné funkce.

### 3.3.3 GitHub

Jak již bylo řečeno v sekci 2.2.1, GitHub není typickým cloudovým úložištěm, lze jej však tímto způsobem s drobnými omezeními využívat. Služba je populární zejména mezi vývojáři, nicméně může se jednat o zajímavou alternativu i pro ostatní uživatele. Díky nástroji Git je totiž tvorba prezentace obohacena o další zajímavou možnost, a sice o pozorování postupné práce při vytváření prezentace.

Vývojář aplikace provede stejně jako u předchozích úložišť registraci nové OAuth aplikace[13] a aplikaci jsou přiděleny Credentials.

Pomocí volání API může vývojář vytvořit repozitář, který bude používán pro potřeby aplikace. Do tohoto repozitáře lze následně nahrávat soubory. V repozitáři nelze samostatně vytvářet složky, ty jsou vždy vytvářeny společně se soubory pomocí určení cesty (vytvoření složky `images` proběhne při nahrání prvního souboru s cestou `images/photo.jpg`). Maximální velikost souboru umístěného v repozitáři je omezena na 100 MB. Kromě velikosti souboru se omezení týkají i maximálního počtu API požadavků za hodinu.

[7] https://console.developers.google.com/
[8] https://developers.google.com/drive/api/v3/about-sdk
[9] Možné MIME typy – https://developers.google.com/drive/api/v3/mime-types
[10] https://www.dropbox.com/developers/apps
[11] https://github.com/dropbox/dropbox-sdk-js
[12] https://dropbox.github.io/dropbox-sdk-js/
[13] https://github.com/settings/developers

Pro práci s API nabízí GitHub vývojářům nástroj Octokit[14]. Ukázky jeho používání se nachází v oficiální dokumentaci[15].

### 3.3.4 GitLab

GitLab je velmi podobným typem nástroje jako GitHub z předchozí sekce. Byl zvolen na základě narůstající členské základny. Díky tomu, že obě služby využívají verzovací systém Git, tak i zde bude možné sledovat průběžnou práci při tvorbě prezentace. Oproti GitHub téměř odpadají omezení ohledně velikosti souboru a počtu požadavků na aplikační rozhraní za hodinu.

Pro vývojáře pracující s jazykem Javascript je připraven nástroj @gitbeaker[16], jež nabízí komplexní možnosti pro práci s API. Používání rozhraní je opět popsáno v přehledné dokumentaci[17].

[14] https://github.com/octokit
[15] https://docs.github.com/en/rest/reference
[16] https://github.com/jdalrymple/gitbeaker
[17] https://docs.gitlab.com/ee/api/api_resources.html

# Kapitola 4

# Návrh

První část (4.1) této kapitoly se zaměřuje na uživatelské rozhraní. Jsou zmíněny nástroje, kterými je návrh inspirován, popsán celkový koncept, ukázány podklady a konečný vzhled aplikace. Sekce 4.2 ukazuje strukturu složek prezentace a popisuje princip ukládání informací o prezentaci do konfiguračního souboru. Oddíl 4.3 se věnuje návrhu aplikace a jejím základním principům.

## 4.1 Uživatelské rozhraní

Za základní prvky dobrého uživatelského rozhraní lze označit jednoduché a přehledné ovládání, rychlý průchod rozhraním a dobře zvolený kontrast barev. Všechny tyto vlastnosti byly při tvorbě grafického návrhu zohledněny.

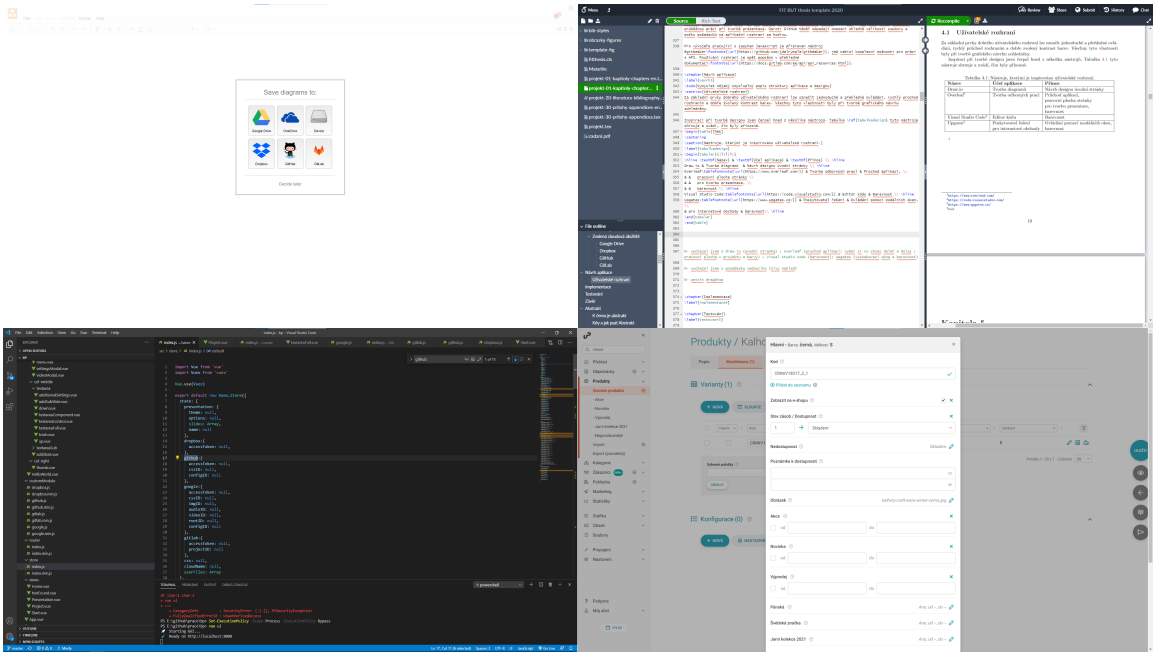
Při tvorbě designu jsem se nechal inspirovat hned několika nástroji. Tabulka 4.1 nástroje shrnuje a uvádí, čím byly přínosné.

Tabulka 4.1: Nástroje, kterými je inspirováno uživatelské rozhraní.

| **Název** | **Účel aplikace** | **Přínos** |
|---|---|---|
| Draw.io | Tvorba diagramů | Návrh úvodní stránky |
| Overleaf[1] | Tvorba odborných prací | Průchod aplikací, pracovní plocha stránky pro tvorbu prezentace, barevnost |
| Visual Studio Code[2] | Editor kódu | Barevnost |
| Upgates[3] | Poskytovatel řešení pro internetové obchody | Ovládání pomocí modálních oken, barevnost |

Každý z těchto nástrojů se těší vysoké popularitě uživatelů, z čehož jsem usoudil, že jejich prostředí budou zpracována dobrým způsobem. Obrázek 4.1 ukazuje prostředí jednotlivých aplikací.

[1] https://www.overleaf.com/
[2] https://code.visualstudio.com/
[3] https://www.upgates.cz/



Obrázek 4.1: Prostředí nástrojů Draw.io, Overleaf, Visual Studio Code a Upgates

Na obrázku 4.2 lze vidět návrhy jednotlivých stránek (jejich popis následuje dále v textu), obrázek 4.3 zachycuje výslednou podobu jedné ze stránek.

Když přijde uživatel poprvé do aplikace, je mu nabídnut výběr cloudových úložišť, do kterých lze prezentace ukládat. Na další stránce dostane dvě možnosti, a sice vytvořit novou nebo otevřít již dříve vytvořenou prezentaci.

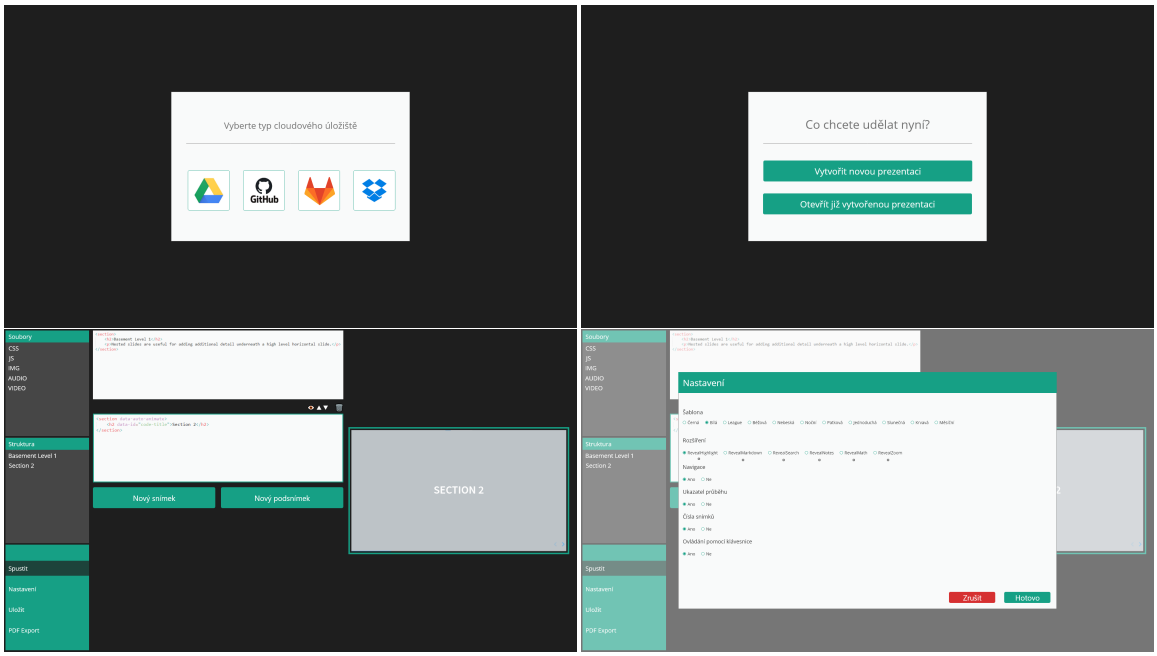
Hlavní stránka pro tvorbu prezentace se člení do tří částí. První z nich – menu – je určeno k ovládání aplikace. V jeho horní části se nachází sekce určená pro práci se soubory. Každé z tlačítek otevírá modální okno. V modálním okně pro CSS se nachází textový editor, kde uživatel může psát vlastní kaskádové styly a ty následně použít v prezentaci. Okna pro obrázky, video a audio mají stejnou strukturu. V horní části se nachází prostor k nahrávání souborů (pomocí metody drag&drop), pod ním pak seznam všech souborů patřící do dané kategorie. Spodní část menu je určena ke spouštění, ukládání a nastavení prezentace. V nastavení se volí jedna z výchozích šablon a další dodatečné možnosti ovládání prezentace (pozice navigačních šipek, zobrazení čísel snímků, . . . ).

Středový panel je místem pro vytváření samotné prezentace. Každé formulářové pole představuje jeden snímek prezentace. Snímky lze dle potřeby přesouvat, mazat, doplňovat o atributy a rozšiřovat o podsnímky – to vše pomocí tlačítek umístěných v pravé horní části každého snímku. Pro přidání nového snímku slouží velké tlačítko plus umístěné vždy za posledním formulářovým polem.

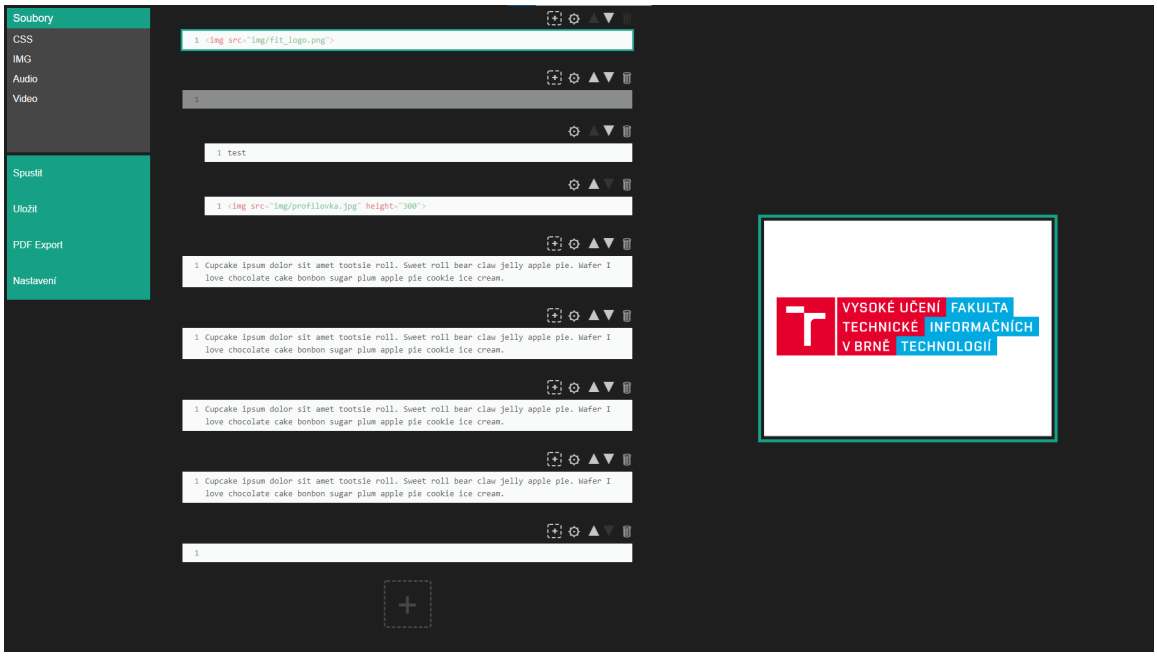
V pravé části stránky se nachází živý náhled aktuálně upravovaného snímku. V případě, že by uživatel potřeboval náhled zvětšit na celou obrazovku, může na něj dvakrát kliknout.

Výsledná podoba uživatelského rozhraní (obrázek 4.3) doznala několika změn. Tou nejvýraznější je zrušení sekce *Struktura* — všechny snímky nemusí nutně obsahovat nadpis, podle kterého měl být tvořen obsah této sekce. Dále bylo nutné změnit koncept tlačítek pro přidání nového snímku a podsnímku — tlačítka by totiž musela „skákat“ dle aktuálně aktivního snímku. Zůstalo jen jedno velké tlačítko pro přidání nového snímku a tlačítko pro přidání podsnímku se přesunulo ke každému snímku vedle tlačítek pro pohyb. Díky

tomu, že Vue dokáže okamžitě reagovat na změnu hodnoty formulářových polí, mohla být zrušena také tlačítka *Hotovo* a *Zrušit* v modálním okně pro nastavení. Došlo také k přidání načítací animace, která se objevuje vždy, když probíhá nějaká komunikace s úložištěm. Uživatel je lépe informován o probíhající akci a aplikace nepůsobí dojmem, že zamzrla. S postupem vývoje byly přidávány do rozhraní další prvky, na které nebyl brán zřetel při tvorbě grafického návrhu. Budou popsány v kapitole 5.




Obrázek 4.2: Výchozí grafický návrh aplikace – úvodní obrazovka, vytvoření/otevření prezentace, prostředí pro tvorbu prezentace a modální okno




Obrázek 4.3: Výsledná podoba stránky pro tvorbu prezentace

## 4.2 Ukládání prezentací do úložiště

Ukládání prezentací do úložiště je řešeno následujícím způsobem. Všechny prezentace budou umístěny v kořenové složce `BPFIT`. Ta bude automaticky vygenerována každému člověku používajícímu aplikaci. Při vytvoření bude mít každá z prezentací tuto strukturu:

```
/nazev-prezentace
├── /css
│   └── style.css
├── /img
├── /video
├── /audio
└── config.json
```

Základ tvoří kořenová složka nesoucí název prezentace. Z názvu je odebrána diakritika, případné mezery nahrazují pomlčky. Obsahuje složky pojmenované dle zaužívaných konvencí – `css`, `img`, `video`, `audio`. Ve složce `css` je vždy umístěn soubor `style.css` určený pro kaskádové styly uživatele. Zbylé složky představují prostor pro média nahraná uživatelem. V případě úložišť Google Drive a Dropbox jsou prázdné, při použití GitHub a GitLab obsahují prázdný soubor `.gitkeep`. Nejdůležitějším prvkem je soubor `config.json`.

### 4.2.1 Soubor config.json a jeho struktura

Soubor obsahuje všechna data týkající se dané prezentace. Stejně jako `style.css` je generován vždy při vytvoření nové prezentace. Každý soubor `config.json` má tuto strukturu:

```
{
    "name": "název prezentace",
    "theme": "black",
    "options": {
        "controls": true,
        "controlsLayout": "bottom-right",
        ...
    },
    "slides": [
        {
            "position": 0,
            "code": "<p>text</p>",
            "attrs": [...],
            "subSlides": [...]
        }
    ]
}
```

Položka `name` představuje název prezentace (tentokrát bez úprav). V `theme` je uložen název jedné z šablon, které poskytuje Reveal.js (ve výchozím stavu `black`). `Options` obsahuje uživatelem specifikované nastavení prezentace. Jedná se o již zmiňovanou možnost nastavení pozice šipek, zobrazení čísel snímků a další[4]. Ve výchozím stavu jsou tyto možnosti

[4] https://github.com/hakimel/reveal.js/blob/master/js/config.js

nastaveny podle frameworku Reveal.js. Pole objektů `slides` obsahuje data jednotlivých snímků. U každého z nich se ukládají následující informace:

- pozice – pozice snímku v rámci prezentace (aplikace umožňuje přesouvání snímků),
- kód – obsah snímku zapsaný v HTML či Markdown,
- pole atributů – jednotlivé atributy nastavující vzhled nebo chování snímku,
- pole podsnímků.

Reveal.js umožňuje pouze jednu úroveň zanoření snímků. U podsnímku tedy stačí ke každému ukládat už jen pozici, kód a atributy.

## 4.3 Aplikace

Základní osu aplikace tvoří čtyři stránky a čtyři moduly pro komunikaci s úložišti. Díky možnostem používaných technologií není třeba pracovat s žádnou databází, řešit lokální nebo dočasné ukládání souborů ani vytvářet vlastní serverové skripty. Veškerý tok se odehrává čistě jen mezi aplikací a cloudovým úložištěm.

### 4.3.1 Trvale udržovaná data

Sdílení dat napříč aplikací umožňuje Vuex. Pro zjednodušení bude místo, kde dochází k uchovávání dat, dále v textu označováno jako „Sklad“.

Každé ze zvolených úložišť má lehce odlišné potřeby, na jejichž základě mohou být do skladu ukládána ID souborů a složek. Vždy se však uchovávají informace o přístupovém kódu, aktuálně upravované prezentaci, vlastních kaskádových stylech, názvu zvoleného úložiště a seznam nahraných médií. Blíže se celé problematice bude věnovat sekce 5.2.

### 4.3.2 Moduly pro úložiště

Pro každé z úložišť je vytvořen modul. Každý z nich staví na jednom z nainstalovaných npm balíčků. Modul obsahuje potřebné funkce pro komunikaci s úložištěm. Cíle a logika funkcí jsou v rámci všech modulů velmi podobné — vše se jen lehce přizpůsobuje danému úložišti.

Moduly jsou importovány do komponent v aplikaci a jejich funkce se volají z `methods` nebo `watch` s patřičnými argumenty. Volané funkce buď vracejí do komponent (či skladu) získaná data nebo vyvolávají zobrazení hlášky o provedené akci (*Uloženo*).

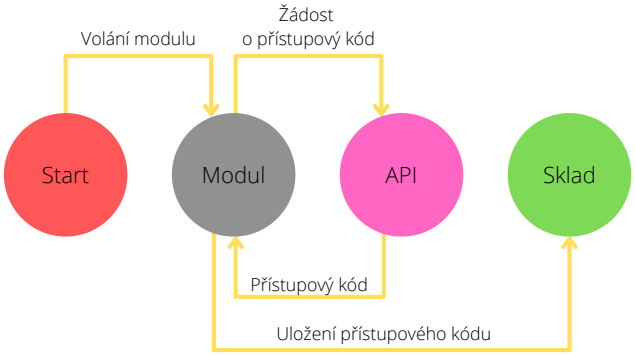
Používané npm balíčky pak v sobě obsahují prostředky pro vytváření požadavků na aplikační rozhraní (popsáno v sekci 2.2.2).

### 4.3.3 Domovská stránka

Jedinou úlohou domovské stránky je poskytnout uživateli výběr cloudového úložiště. Po výběru je volána patřičná funkce z modulu, která uživatele přesměruje na stránku organizace, kde provede přihlášení. Po přihlášení proběhne automatické přesměrování zpět do aplikace na stránku *Start*.

### 4.3.4 Stránka Start

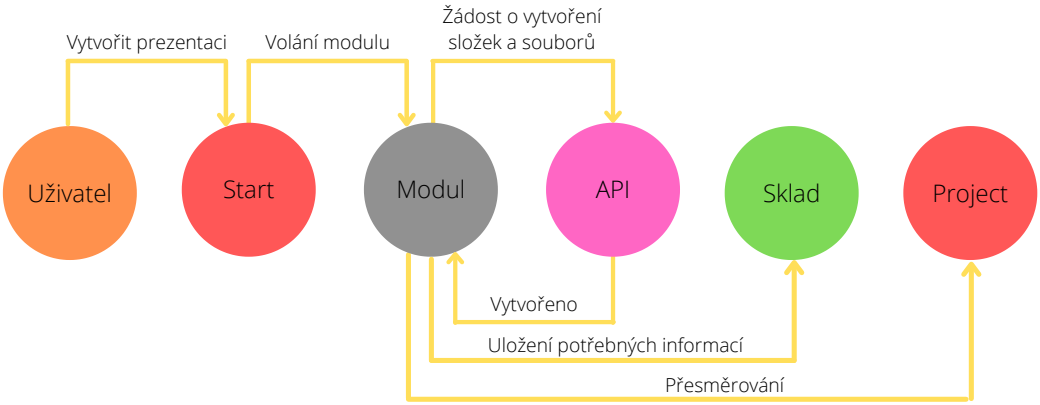
Při prvním příchodu na stránku *Start* aplikace přečte z URL parametry `code` a `source`, v němž se nachází informace o zvoleném úložišti. Parametr `code` je okamžitě zasílán v těle požadavku na vytvoření přístupového kódu (dokončení autorizace). Vrácený `access_token` se společně s druhým z parametrů ukládají do skladu.




Obrázek 4.4: Tok událostí při načtení stránky *Start*.

Následně má uživatel možnost vytvořit nebo otevřít prezentaci. V případě vytváření zadá název a v úložišti je nové prezentaci vytvořena složka (popsáno v 4.2), do skladu se ukládá výchozí nastavení prezentace. Otevírání prezentace probíhá ve dvou fázích. V první dojde k zavolání funkce z modulu, která získá všechny prezentace uživatele, v druhé následuje otevírání vybrané prezentace a načtení jejích dat do skladu. Po vytvoření nebo otevření směrovač přesune uživatele na stránku *Project*, která slouží k editaci prezentace.

Při dalších příchodech na stránku *Start* (uživatel chce otevřít jinou prezentaci) jsou ze skladu vyprazdňovány informace o naposledy otevřené prezentaci.




Obrázek 4.5: Tok událostí při vytvoření prezentace.

### 4.3.5 Stránka pro editaci prezentace – Project

Nejpodstatnější stránka celé aplikace *Project* je místem, kde probíhá tvorba prezentace. Skládá se z mnoha spolupracujících komponent (obrázek 4.6), z nichž každá má svou logiku a volá různé funkce z modulu. Jejich bližší účel a zpracování bude podrobněji popsáno

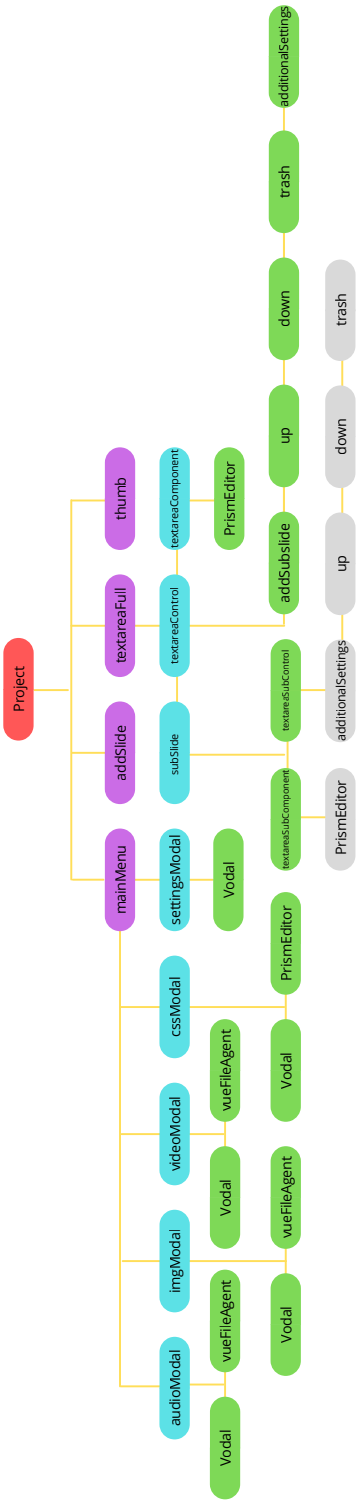
v kapitole 5. Při příchodu jsou načítána data ze skladu do dat jednotlivých komponent. Rodičovská komponenta stránky sama o sobě uchovává tato data:

- `stateMsg` – pro zobrazování hlášek (*Uloženo*),
- `showLoading` – pro podmíněné zobrazování načítací animace,
- `counter` – pro potřeby přidávání snímků,
- `textareas` – snímky,
- `currentCode` – kód aktuálně upravovaného snímku,
- `currentAttrs` – atributy aktuálně upravovaného snímku,
- `options` – nastavení prezentace,
- `theme` – zvolená šablona,
- `startSaveFlag` – pro informování subkomponent o začátku ukládání,
- `timeoutID` – plánování automatického ukládání.

Na základě stavu skladu se při načtení stránky automaticky generuje potřebný počet polí pro snímky, nastavuje se jejich hodnota, atributy, podsnímky… S těmito snímky lze dále pracovat pomocí metod v komponentě obsažených (`addTextarea`, `move`, …). Rodičovská komponenta je jakousi spojkou pro komponenty nižší úrovně, které mezi sebou potřebují komunikovat a předávat si informace. Příkladem takového chování je vytváření náhledu — uživatel vepíše svůj kód do formulářového pole, ten je předán rodičovské komponentě, ta jej dále posouvá do komponenty `thumb`, která náhled vygeneruje.

Mezi prvky hlídané pomocí `watch` se řadí kód a atributy aktuálního snímku, šablona a nastavení prezentace. Při změně některého z těchto prvků aplikace plánuje automatické ukládání, které proběhne v případě, že uživatel do pěti minut neprovede sám uložení.

Při stisku tlačítka *Spustit* nebo *PDF Export* se provede uložení aktuálního stavu prezentace do skladu. Následně je uživatel přesměrován na stránku *Presentation*, kde vidí kompletní výsledek své práce.




Obrázek 4.6: Struktura komponent stránky *Project*. Červená barva představuje nejvýše postavenou komponentu, fialová znázorňuje její bezprostřední následníky, kteří pomáhají se strukturováním. Níže postavené komponenty zastupují jednotlivá formulářová pole (snímky), různá tlačítka, modální okna, prostor pro nahrávání souborů a jiné.

### 4.3.6 Stránka pro prezentování – Presentation

Při příchodu na tuto stránku dojde k přečtení dat ze skladu. Následně dochází k seřazení snímků dle uživatelem požadované pozice. Po seřazení začíná transformace dat do HTML kódu, jehož struktura je dána požadavky frameworku Reveal.js. Po dokončení transformace se tento kód vloží do stránky — konkrétně do tagu `div` s id `slides`. Poslední akcí, která musí být provedena před tím, než uživatel začne prezentovat, je inicializace samotného frameworku Reveal.js. Ten dává prezentaci její finální podobu.

Stránka pro prezentování a export do PDF sdílí společný základ. Rozdíl spočívá v přidání parametru `print-pdf` do URL adresy v případě exportu. Jakmile Reveal.js zjistí, že tento parametr se v adrese nachází, tak uzpůsobuje své chování a generuje snímky, které mají vhodný formát k uložení. S tímto parametrem pracuje i sama aplikace — automaticky otevírá dialogové okno pro tisk. Jediným úkolem uživatele zůstává, aby jako cíl tisku nebyla nastavena tiskárna, ale možnost *Uložit do PDF*. Následně je mu do jeho počítače stažen PDF soubor s názvem zadaným při vytváření prezentace.

# Kapitola 5

# Implementace

Kapitola se věnuje popisu implementace různých částí aplikace. První dvě sekce objasňují směrování a uchovávání dat v rámci aplikace. Další se již zaměřují na principy jednotlivých úkonů, které lze v rámci aplikace provádět a říkají, co se zrovna odehrává v pozadí. Pro názornou demonstraci je využíváno krátkých výpisů kódu.

## 5.1 Směrování

O správné směrování se stará Vue router[1]. Všechny možné adresy[2] jsou zaneseny jako objekty v poli `routes`. Každý objekt obsahuje položky `path` (cesta), `name` (pojmenování cesty) a `component` (nejvýše postavená komponenta v rámci cesty). Toto pole je používáno jako jeden z parametrů při inicializaci směrovače.

Navigace je v aplikaci realizována pomocí `<router-link>` (Vue.js ekvivalent k tagu `<a>`) nebo pomocí příkazů `this.$router.push( path: '/' )` a `this.$router.go(-1)`.

```
const routes = [
  {
    path: '/',
    name: 'Home',
    component: Home
  }
]
```

```
const router = new VueRouter({
  mode: 'history',
  routes
})
```

Výpis 1: Ukázka inicializace směrovače

## 5.2 Sdílený stav aplikace

Ve skladu se nacházejí čtyři objekty – `state`, `mutations`, `actions` a `getters` – sloužící pro lokální uchovávání dat, respektive pro manipulaci s nimi. `State` obsahuje objekt `presentation`, do kterého se ukládají data aktuálně upravované prezentace. Jeho struktura je shodná se strukturou souboru `config.json` 4.2.1. Kromě `presentation` se ve `state` nachází pole `userFiles`, kam se ukládá seznam nahraných médií, položky `css` a `cloudname`

[1] Cesta k souboru – `/src/router/index.js`
[2] Domovská stránka, Start, Project, Presentation, Stránka nenalezena

pro kaskádové styly, respektive název zvoleného úložiště a objekty pojmenované podle jednotlivých úložišť, do nichž se ukládá přístupový kód a případné identifikátory složek a souborů. Díky ukládání identifikátorů se snižuje počet vytvářených požadavků na aplikační rozhraní úložiště.

V objektu `mutations` se nachází funkce, které mohou přistupovat k položkám z objektu `state` a měnit jejich hodnoty. Tyto funkce jsou volány z `actions`. Pro získání hodnoty položky z objektu `state` slouží funkce z objektu `getters`.

```
state: {
    presentation: {
      theme: null,
      options: null,
      slides: Array,
      name: null
    },
    github:{
      accessToken: null,
      cssID: null,
      configID: null
    },
    gitlab:{
      accessToken: null,
      projectID: null
    },
    ...
}
```

```
mutations: {
    changeTheme(state, theme) {
      state.presentation.theme = theme
    },
    ...
},
actions: {
    changeTheme(context, theme){
      context.commit('changeTheme', theme)
    },
    ...
},
getters:{
    getPresentationData: state => {
      return state.presentation
    },
    ...
}
```

Výpis 2: Objekty `state`, `mutations`, `actions` a `getters` ve skladu

Příkaz `this.$store.dispatch('changeTheme', this.theme)` zavolá funkci pro změnu šablony, které je předána aktuálně nastavená hodnota. Hodnota je přiřazena k položce `theme` objektu `presentation`. Naopak k získání všech dat aktuální prezentace slouží příkaz `this.$store.getters.getPresentationData`. Tento přístup se využívá k manipulaci se všemi daty uchovávanými ve skladu.

## 5.3 Volba úložiště

Při kliku na tlačítko vybraného úložiště se z metod komponenty `cloudButton` volá funkce `login()`. V ní je rozlišeno o jaké úložiště se jedná. Následně je zvolen patřičný modul[3] a volána funkce, která zařídí přesměrování na stránku organizace (například `gitlabAuthUrl()`). Funkce sestaví URL pro přesměrování s patřičnými parametry (více v sekci 2.2.3). Důležitým prvkem je specifikace parametru `source` v návratové adrese (ta je rovněž parametrem volaného URL). Jedná se o vlastní parametr, který v sobě nese název zvoleného úložiště. Díky němu je pak aplikace na stránce *Start* schopna zjistit, jaké úložiště uživatel v předchozím kroku zvolil. Následuje přesměrování na stránku organizace pomocí funkce `open()`.

[3] Zdrojové soubory všech modulů se nachází ve složce `/src/customModule/`

## 5.4 Získání přístupového kódu

V bodě `mounted` stránky *Start* se volá funkce (například `gitlabAccessToken()`) z modulu. V ní dochází k vytvoření požadavku pro získání přístupového kódu na aplikační rozhraní zvoleného úložiště.

Požadavek je vytvořen pomocí modulu `request`, metoda `POST` jej zasílá na patřičný koncový bod[4]. V těle požadavku se odesílá `code`[5], `client_id`, `client_secret` a další parametry v závislosi na úložišti. Hodnoty vkládané do `client_id` a `client_secret` jsou uchovávány v souboru `.env`.

Po odeslání požadavku je autorizačním serverem vrácena odpověď ve formátu JSON. V případě úspěchu vrácený objekt obsahuje položku `access_token`. Její hodnota je ukládána do skladu a využívána dále při jakékoliv komunikaci s úložištěm.

## 5.5 Vytvoření nové prezentace

Při vytváření nové prezentace dochází nejprve k zobrazení formulářového pole (podmíněné zobrazení na základě hodnoty proměnné `newPresentation`), kam uživatel zadá požadovaný název prezentace. Kliknutí na tlačítko *Potvrdit* volá metodu `savePresentationName()`. V ní se pomocí regulárního výrazu nejprve kontroluje, zda uživatel vložil validní název prezentace (za špatně vložený název se považuje například řetězec pouze z bílých znaků). Název prezentace, který prošel kontrolou, se uloží do skladu a následně je volána funkce jednoho z modulů, která skrze aplikační rozhraní vytvoří základní soubory a složky (popsáno v 4.2) ve zvoleném úložišti.

V takové funkci nejprve dochází ke zjištění, zda existuje složka `BPFIT`, případně k jejímu vytvoření. Následně se v ní vytváří složka pro prezentaci (pouze v případě, že prezentace s daným názvem ještě neexistuje), ta je pojmenována podle vloženého názvu prezentace, z nějž je odebrána diakritika (pomocí `latinize`) a mezery nahrazeny pomlčkami. V takto přichystané složce už mohou být generovány složky pro kaskádové styly a média a soubor `config.json` obsahující název prezentace a její výchozí nastavení. Po dokončení tohoto procesu je uživatel přesměrován na stránku *Project*.

S ohledem na specifické požadavky úložišť mohou být do skladu během vytváření prezentace ukládány doplňující data potřebná pro nahrávání či aktualizaci souborů. Jedná se o identifikátory:

- souborů `style.css` a `config.json`,
- složek `css`, `audio`, `img` a `video`,
- složky prezentace.

Tato data aplikace získává v odpovědi vrácené aplikačním rozhraním úložiště při vytváření souboru/složky. Do skladu se ukládají do objektu pojmenovaném podle úložiště.

## 5.6 Otevření prezentace

Na začátku procesu otevření prezentace stojí metoda `showUserPresentations()`. Ta zavolá funkci z modulu. Zde dochází k vytvoření požadavku na aplikační rozhraní, jehož cílem

[4] Například v případě GitLab – https://gitlab.com/oauth/token

[5] Tuto hodnotu aplikace přečetla z URL po té, co uživatel provedl přihlášení a byl přesměrován zpět.

je získat obsah složky `BPFIT`. Rozhraní v odpovědi vrací objekt, z kterého funkce vyfiltruje pouze složky (pro případ, že by uživatel ručně do složky umístil nějaký soubor). Údaje o jednotlivých složkách se ukládají do pole `userPresentations` v komponentě. Pokud pole obsahuje alespoň jednu položku, tak je uživateli zobrazen seznam prezentací.

Kliknutím na požadovanou prezentaci uživatel spustí metodu `openPresentation()`. Opět dochází k volání modulu a jeho funkce, které je předán název prezentace (v závislosti na úložišti případně další identifikátor), na jehož základě jsou pomocí API získávány požadovaná data. Dojde k získání seznamu nahraných médií a načtení obsahů souborů `style.css` a `config.json`. Získaná data se postupně ukládají do pole `userFiles`, proměnné `css` a objektu `presentation` ve skladu. Po dokončení je uživatel přesměrován na stránku *Project*. Příslušné komponenty si data uložená ve skladu převezmou a postarají se o jejich zobrazení uživateli.

## 5.7 Přidání atributu ke snímku

Přidání atributu ke snímku probíhá v komponentě `additionalSettings`. Do formulářového pole uživatel vepíše požadovaný atribut (například `data-background-color="red"`) a stiskne klávesu Enter, čímž dojde k volání metody `onEnter()`.

Pomocí regulárních výrazů v kombinaci s funkcí `replace()` jsou z vloženého řetězce odebrány veškeré uvozovky. Následuje rozdělení slova podle znaku „=“. Po rozdělení dochází k odebrání bílých znaků. Před znakem „=“ jsou odebrány veškeré bílé znaky, z druhé části všechny na začátku a na konci, případných více mezer mezi slovy je nahrazeno jednou mezerou. Kromě regulárních výrazů a `replace()` je v tomto případě využívána i funkce `trim()`.

První část nesmí být `id`. Pokud první část odpovídá řetězci `class`, tak dochází ke kontrole druhé části, která musí obsahovat validní název jedné nebo více tříd[6] (opět pomocí regulárních výrazů). Jiné názvy a hodnoty atributů jsou ošetřeny dále. V případě, že uživatelem vložený atribut neobsahuje „=“ (například `data-markdown`), tak je na druhou část nahlíženo jako na prázdný řetězec.

Finální kontrolu atributu provádí funkce `testAttributes()`. Šablona komponenty obsahuje element `<section>`. Na tento element jsou atributy aplikovány také v rámci prezentace. Toho lze využít k otestování uživatelem vloženého atributu ještě před tím, než dojde k předávání atributu mezi komponentami a přidání do prezentace. Pomocí funkce `setAttribute()` a konstrukce `try{...} catch{...}` je vyzkoušena validita atributu.

V posledním kroku je kontrolováno, zda daný atribut již nebyl dříve vložen. Pokud všechny kontroly proběhnou úspěšně, tak dojde k přidání atributu do prezentace. V opačném případě uživatel získá informaci, která z kontrol zastavila vkládání atributu.

## 5.8 Komunikace mezi komponentami

Předávání dat a volání funkcí mezi komponentami je možno realizovat třemi způsoby. První z nich je využíván například v případě přidání nového snímku. V metodě `addTextarea()` komponenty `addSlide` je volán příkaz `this.$parent.addTextarea()`. Tento příkaz zavolá stejnojmennou metodu z rodičovské komponenty `Project`, která zajistí přidání nového snímku. Je možné zřetězit více `$parent` za účelem překlenutí více úrovní.

[6] https://www.geeksforgeeks.org/which-characters-are-valid-in-css-class-names-selectors/

Druhou možnost představuje vyvolání události. Tato možnost je využívána při změně obsahu snímku. Když uživatel vepíše znak do formulářového pole představující snímek, tak dojde v rámci komponenty `textareaComponent` k volání funkce `readVal()`. Ta pomocí příkazu `this.$emit("toParent", this.code)` vyvolá událost `toParent` a odešle nově vepsanou hodnotu kódu. Rodičovská komponenta `textareaFull` tuto událost pomocí direktivy `v-on` odchytne a může dále pracovat s předanou hodnotou kódu.

První dvě možnosti slouží ke komunikaci zdola nahoru. Pokud však potřebuje rodičovská komponenta předat data níže postavené komponentě, tak využije direktivu `v-bind`. Hodnotu pak přečte subkomponenta v `props`. Tohoto chování aplikace využívá pro předání kódu do komponenty `thumb`, která z něj následně vygeneruje jeden snímek prezentace.

## 5.9 Vložení vlastních kaskádových stylů

Úprava CSS probíhá v komponentě `cssModal`. Kaskádové styly se pomocí direktivy `v-model` průběžně zapisují do položky `cssString` v objektu vraceném funkcí `data()`. Po kliknutí na tlačítko uložit je volána metoda `processCSS()`. Zde probíhá kontrola syntaxe vložených stylů pomocí balíčku csstree-validator. V případě nesprávného zápisu je uživatel vyzván k nápravě. Pokud kontrola proběhne úspěšně, dojde k uložení vložených stylů do skladu – realizováno příkazem `customThis.$store.dispatch('setCSS', customThis.cssString)`.

Následně se pomocí funkce `loadCSSToBrowser()` načtou definované styly do prohlížeče. Před samotným načtením probíhá modifikace vloženého CSS. Vložený řetězec je obalen řetězcem `.slides{}`, což je jedna ze tříd, kterou obsahuje tag `<div>` nadřazený všem snímkům prezentace. Díky tomuto obalení vzniká takzvaný SCSS zápis. Takto vytvořené SCSS je následně kompilováno zpět do CSS, které už lze bezpečně načíst do prohlížeče. Nejprve dojde k vytvoření elementu `style` pomocí `document.createElement('style')`, následně je nastaven jeho obsah pomocí `innerHTML`, nakonec dojde k vložení elementu do hlavičky stránky – `document.head.appendChild(userStyle)`. Transformace do SCSS a zpět do CSS je nutná z toho důvodu, že styly definované uživatelem jsou vkládány přímo do dokumentu. V případě, že by uživatel definoval styl příliš obecně (například pouze `a{...}`), mohlo by dojít k narušení prostředí celé aplikace. Transformace zajistí, že tyto styly budou aplikovány pouze v rámci `div` s třídou `slides`.

Posledním krokem v rámci celého procesu přidání vlastních stylů je jejich uložení do cloudového úložiště. Dochází k volání funkce některého z modulů (například `gitlabCSSSave()`). Funkci je předán původní řetezec se styly. Vytváří se požadavek na API, který aktualizuje obsah souboru `style.css` v úložišti.

```
.slides{
    a{
        color: red;
    }
}
```

```
.slides a{
    color: red;
}
```

Výpis 3: Princip transformace SCSS do CSS

## 5.10 Nahrávání souborů

Soubory lze nahrávat v komponentách `audioModal`, `imgModal` a `videoModal`. Velkou část práce při nahrávání odvádí komponenta `VueFileAgent` (vycházející z balíčku vue-file-agent), jejíž chování je definováno pomocí atributů. Mezi významné patří:

- `multiple` – zákaz (nebo povolení) nahrávání více souborů najednou,
- `accept` – omezení typu souboru (například `image/*`),
- `maxSize` – maximální velikost souboru.

U vybraného souboru dochází ke kontrole, zda splňuje definované požadavky a k umístění do pole `fileRecordsForUpload`. Zde se nachází všechny potřebné informace pro další práci se souborem (název souboru, typ souboru, velikost, obsah, ...). Následně stiskem tlačítka *Uložit* dojde k volání funkce z modulu, která zajistí nahrání do cloudového úložiště. Toto chování může například zajišťovat `dropBoxIMGSave()`. Zde dojde nejprve k sestavení cesty nahrávaného souboru[7]. Následně funkce `filesUpload()` z balíčku dropbox vytvoří požadavek na API a dojde k nahrání souboru do úložiště. V požadavku je předávána vytvořená cesta a obsah souboru.

Po nahrání do úložiště dojde k získání seznamu všech souborů ve složce určené typem média (tedy v případě nahrání obrázku dojde k získání seznamu všech obrázků ze složce `img`). Seznam těchto souborů se následně zobrazí v komponentě pod nahráváním. Odsud je možné vyvolat odstranění souboru z úložiště (například pomocí `dropBoxDeleteImage()`).

## 5.11 Automatické návrhy souborů

Pole `userFiles` ve skladu obsahuje informace o všech nahraných souborech. Tato data se využívají k návrhu souborů v komponentách `textareaComponent` a `textareaSubComponent`. Když uživatel napíše do formulářového pole například řetězec `<img src="">` a kurzor se nachází mezi uvozovkami, tak pod formulářovým polem dojde k zobrazení seznamu všech nahraných souborů ve složkách `audio`, `img` a `video`. Pozice kurzoru ve formulářovém poli je kontrolována pomocí `selectionStart`, okolí pomocí regulárních výrazů a funkce `find()`. Pokud uživatel pokračuje ve psaní mezi uvozovkami, tak dojde k procházení `userFiles`. Každý záznam je porovnáván pomocí funkce `startsWith()` oproti řetězci nacházejícímu se v uvozovkách. Pokud se začátky obou řetězců shodují, tak tento záznam i nadále zůstává zobrazen, v opačném případě dochází k redukci.

Následně klikem na požadovaný soubor dojde ke vložení řetězce, který obsahuje cestu zvoleného souboru, mezi uvozovky.

## 5.12 Spuštění prezentace

Spuštění prezentace začíná kliknutím na tlačítko *Spustit*. O této události je informována nejvýše postavená komponenta `Project`, která v tuto chvíli spouští metodu `startSave()`. Ta v první řadě uloží do dat prezentace uchovávaných ve skladu aktuálně zvolenou šablonu a nastavení prezentace. Následně dochází ve skladu k aktualizaci všech snímků, které jsou momentálně v prezentaci vytvořeny (ty však ještě nemusí být v cloudovém úložišti).

[7] například `/prezentace/img/obrazek.jpg`

Všechny snímky jsou ze skladu prvně odstraněny. Následně je negací obrácena hodnota booleovské proměnné `startSaveFlag`. Tato změna je kontrolována pomocí `watch` v níže postavených komponentách určených snímkům. Změna komponentě dává příkaz, že má uložit svůj obsah do skladu. Příkazem `this.$store.dispatch('changeSlides', infoToStore)` je vyvoláno ukládání snímku do skladu. Objekt `infoToStore` obsahuje kód a atributy daného snímku, případně také jednotlivé podsnímky. Po dokončení ukládání snímků dochází k přesměrování na stránku *Presentation*.

Zde dochází nejprve k získání dat prezentace ze skladu. Následně je volána funkce `sort()`, která zajistí seřazení snímků podle požadované pozice. Následně dochází k transformaci dat ze snímků do HTML. Transformace je realizována postupným přidáváním řetězců do proměnné `code`. Pro každý z hlavních snímků je vytvořen rodičovský tag `<section>` doplněný o unikátní `id`. Obsahem tohoto prvku je kód zapsaný uživatelem do formulářového pole. V každém kódu je kontrolováno, zda není přítomen řetězec, který by představoval cestu k uživatelem nahranému mediálnímu souboru. Pokud se takový řetězec podaří v kódu nalézt, tak je nahrazen absolutní URL adresou odkazující na daný soubor v úložišti. Případné podsnímky procházejí stejným procesem a jsou zařazeny do obsahu rodičovského tagu `<section>`. Takto transformovaný kód je vložen do dokumentu pomocí `document.getElementById('slides').innerHTML = code`.

Následně jsou jednotlivé snímky (a podsnímky) doplňovány o uživatelem specifikované atributy. Požadovaný tag `<section>` je vyhledán pomocí `id` a atribut nastaven funkcí `setAttribute()`. Pokud uživatel specifikoval například atribut `data-background-image` s hodnotou představující cestu k nahranému obrázku, tak rovněž dochází k nahrazení této hodnoty absolutní URL adresou souboru.

Volání metody `callReveal()` inicializuje framework Reveal.js. Po inicializaci je prezentace plně připravena.

```
<div class="slides">
    <section id="topLevel0" data-background-color="red">
        <section id="subLevel_0_0">
            <h1>Prezentace</h1>
        </section>
        <section id="subLevel_0_1">
            <p>Cupcake ipsum dolor sit amet topping halvah.</p>
        </section>
    </section>
    <section id="topLevel1">
        <img src="https://www.dropbox.com/s/lxqev3o5n/img.jpg">
    </section>
</div>
```

Výpis 4: Ukázka transformovaného HTML kódu

## 5.13 Ukládání prezentace

Ukládání prezentace začíná stiskem tlačítka *Uložit*, což v nejvýše postavené komponentě `Project` vyvolá událost `startSaveForCloud` a dojde ke spuštění metody `startSaveCloud()` zahajující ukládání. Stejně jako v případě spouštění prezentace se nejprve uloží do skladu název šablony, nastavení a jednotlivé snímky (ukládání snímků opět zahajuje otočení hodnoty

proměnné `startSaveFlag`). Po uložení všech dat do skladu dochází k volání funkce z modulu – například `dropBoxConfigSave()`. V ní nejprve dochází k určení cesty, kam má být soubor konfigurační soubor uložen. Po určení cesty dojde k volání funkce `filesUpload()`, která zajistí aktualizaci obsahu souboru ve službě Dropbox. Pomocí funkce `stringify()` dojde k převedení JSON objektu obsahujícího data (uloženého ve skladu) prezentace na řetězec, ten je funkci `filesUpload()` předán jako parametr představující obsah souboru. Následně dochází k aktualizaci souboru. O výsledku ukládání aplikace informuje uživatele hláškou. Tato logika se uplatňuje napříč všemi úložišti.

Automatické ukládání se zahajuje v následujících případech:

- u některého ze snímků ze snímků se změní kód nebo atributy,
- prezentaci je změněna šablona,
- prezentaci je změněno nastavení,
- dojde k přidání, odstranění nebo pohybu snímku.

V prvních třech případech jsou tyto události sledovány pomocí `watch` v komponentě `Project` a volají metodu `autoSave()`. Ve čtvrtém případě je metoda pro automatické ukládání volána přímo z metody pro danou akci (metoda pro přidání snímků volá metodu pro automatické ukládání).

Metoda `autosave()` nejprve zkontroluje, zda již nebylo dříve naplánováno automatické uložení. Pokud ano, tak toto ukládání zruší (pomocí `clearTimeout()`) a naplánuje nové. Plánování je realizováno funkcí `setTimeout()`, která po pěti minutách zavolá metodu `startSaveCloud()`.

## 5.14 Úpravy npm balíčků

Během implementace bylo potřeba upravit některé z balíčků, a sice Reveal.js, gdrive-utils, Vodal a vue-file-agent. Poslední dva zmíněné byly upraveny minimálně. Jednalo se jen o přepsání několika kaskádových stylů. Jejich případná budoucí aktualizace neovlivní funkčnost aplikace. První dva zmíněné však prošly rozsáhlejšími úpravami a v případě budoucího přechodu na vyšší verze musí být úpravy dopracovány.

Plugin RevealMath vkládá do aplikace externí skript[8]. Jelikož v průběhu chodu aplikace je Reveal.js (spolu s ním i tento plugin) inicializován hned několikrát, dochází pokaždé k jeho přidání do aplikace. Zamezit takovému chování je možné následovně. V souboru `math.esm.js`[9] je nutné vkládanému skriptu přidat atribut `id` s hodnotou `mathScript`. Díky této úpravě dokáže aplikace rozpoznat, zda byl skript již do aplikace načten, případně zamezit několikanásobnému vkládání.

V rámci frameworku Reveal.js bylo třeba přepracovat kaskádové styly za účelem dynamické změny šablon. Další důvod k tomuto zásahu představoval fakt, že některé styly byly definovány příliš obecně a ovlivňovaly vzhled samotné aplikace (styly jsou totiž importovány přímo do aplikace). Byly vytvořeny dva nové soubory. První z nich nese název `allThemes.css`[10]. Obsahuje styly všech šablon. Každý ze stylů byl rozšířen o třídu představující název dané šablony (předtím – `.reveal table`, nyní – `.black.reveal table`).

[8] URL skriptu – https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.0/MathJax.js
[9] Cesta k souboru – `/node_modules/reveal.js/plugin/math/math.esm.js`
[10] Cesta k souboru – `/node_modules/reveal.js/dist/theme/allThemes.css`

Druhý z nově vytvořených souborů – `revealEdit.min.css`[11] – je rozšířenou kopií výchozího souboru `reveal.css`. Princip úpravy je zcela stejný jako v případě předchozího souboru, tedy doplnění `.reveal` o třídu s názvem šablony. Toto je třeba provést pro každou ze šablon.

V balíčku gdrive-utils nejprve došlo k odstranění souboru `generateToken.js`[12]. Hlavní soubor – `index.js`[13] – ve výchozím stavu obsahoval jen základní funkce pro komunikaci s API. V rámci souboru byla odstraněna závislost na balíčku `readline`. Došlo k modifikaci některých funkcí a vytvoření dalších, tak aby byly pokryty všechny potřeby aplikace. Jmenovitě se jedná o funkce:

- `GDriveUtil(clientId, ClientSecret, redirect, accessToken)` – upraveny parametry funkce, změněno nastavování hodnot,
- `getFileContent(id, cb)` – čtení souboru,
- `setFolderPermision(id, cb)` – nastavení práv složky,
- `listFiles(listCB)` – nastavení parametru pro volání API (původně byly vraceny i soubory z koše),
- `isExistInFolder(name, cb)` – kontrola existence prezentace s daným názvem,
- `getSubFolders(rootID, listCB)` – seznam složek ve specifikované složce,
- `getAllFolderFiles(rootID, listCB)` – seznam souborů ve specifikované složce,
- `uploadFile(parent, name, mime, content, cb)` – nahrání souboru do úložiště,
- `updateFile(fileID, mime, content, cb)` – aktualizace souboru v úložišti,
- `createSubFolder(parent, name, cb)` – vytvoření složky ve složce.

[11] Cesta k souboru – `/node_modules/reveal.js/dist/revealEdit.min.css`
[12] Cesta k souboru – `/node_modules/gdrive-utils/lib/generateToken.js`
[13] Cesta k souboru – `/node_modules/gdrive-utils/lib/index.js`

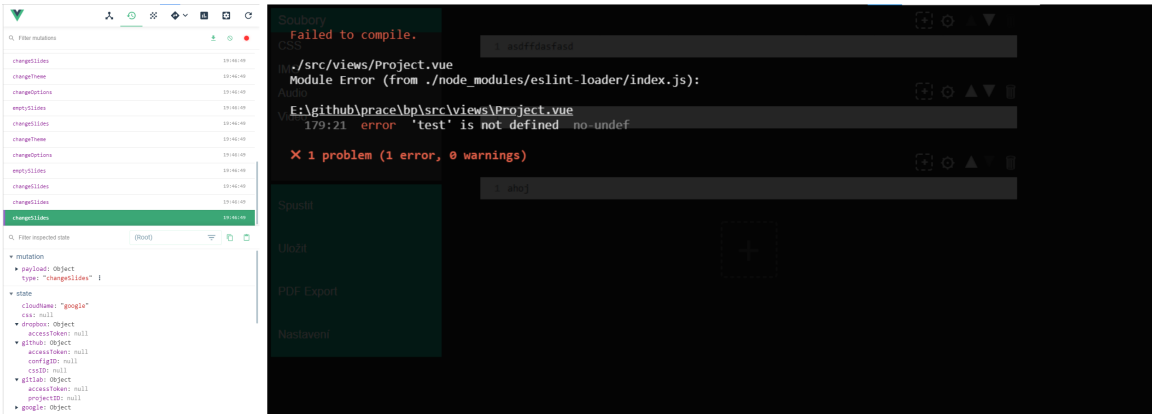
# Kapitola 6

# Testování

Ověřování správné funkčnosti je nedílnou součástí vývoje každého software. Kapitola se věnuje testování, kterým si aplikace prošla. První část (6.1) zmiňuje nástroje, které byly k testování využívány v průběhu implementace. Druhá sekce (6.2) popisuje proces konečného testování. V poslední části (6.3) lze nalézt postřehy uživatelů, kteří s aplikací zkoušeli pracovat.

## 6.1 Testování v průběhu vývoje

V průběhu vývoje byly používány nástroje Vue.js devtools (více v části 3.1.5) a ESLint, který analyzuje kód a dokáže vývojáře upozornit na neznámé nebo nepoužité proměnné, nejednotný styl kódu… Díky tomuto nástroji byla odhalena spousta chyb téměř ihned, nedocházelo k jejich kumulaci a problémy mohly být řešeny okamžitě. Rozšíření Vue.js devtools bylo užitečným pomocníkem hlavně v pozdější fázi vývoje, kdy už aplikace zvládala provádět více provázaných operací a ve skladu docházelo k uchovávání většího množství dat.




Obrázek 6.1: Ukázky nástrojů Vue.js devtools a ESLint. Vlevo je ukázán stav skladu během tvorby prezentace, vpravo nástroj ESLint hlásící chybu.

## 6.2 Konečné testování

Konečné testování proběhlo, po dohodě s vedoucím, formou tvorby netriviální prezentace, během jejíhož vytváření došlo k pokrytí široké funkcionality aplikace, respektive frameworku Reveal.js. Taková prezentace byla vytvořena a následně uložena do každého z podporovaných úložišť. Proces vytváření prezentace zahrnoval:

- změnu šablony na béžovou,
- přesun navigačních šipek na okraje,
- zobrazení čísel snímků,
- nastavení `backgroundTransition` na hodnotu `zoom`,
- vytvoření sedmnácti hlavních snímků,
- vytvoření pěti, dvou a sedmi podsnímků u snímků č. 4, 9 a 16,
- nahrání osmi obrázku (.jpg, .png, .svg, .gif), jednoho audio souboru (.mp3) a dvou videí (.mp4) do úložiště,
- umístění nadpisu `<h1>` a loga FIT VUT (absolutní URL) na úvodní snímek,
- nastavení atributu `data-background-color="red"` u snímku č. 2 snímku,
- využití `fragments` na snímku č. 3,
- vložení ukázky zdrojových kódu do všech podsnímků snímku č. 4,
- vložení videa do snímku č. 5,
- vložení videa na pozadí snímku č. 6,
- vložení odkazů na první, předchozí a následující snímek do snímku č. 7,
- vložení dvou nadpisů `<h2>` obsahujících třídu `r-fit-text` do snímku č. 8
- sazbu matematických znaků ve všech podsnímcích snímku č. 9,
- schování snímku č. 10 pomocí atributu `data-visibility` s hodnotou `hidden`
- vytvoření seznamu s postupným zobrazováním odrážek pomocí Markdown na snímku č. 11,
- využití atributu `data-auto-animate` pro animování obsahu při přechodu mezi snímky č. 12 a 13,
- lokální změnu animace přechodu mezi snímky č. 14 a 15,
- zobrazení obrázků ve všech podsnímcích snímku č. 16 (jednoduché zobrazení, obrázek jako pozadí snímku, využití vlastnosti `r-stack`),
- animace obdélníku pomocí CSS na snímku č. 17,
- prohození snímků č. 7 a 14 a snímků č. 2 a 16.

Jednotlivé kroky byly doprovázeny vkládáním krátkých textů a kombinací různých atributů (například v případě obrázku `data-background-image` a `data-background-size`).

Po vytvoření čtyř prezentací byl zkontrolován obsah složek, `img`, `audio` a `video` a porovnán obsah souborů `style.css` a `config.json`. Očekávaným výsledkem byl stejný obsah všech jmenovaných souborů a složek. Toto se také potvrdilo.

Během tohoto testování se zjistilo, že aplikace neumožňuje přidání atributu, který by měl více hodnot (například `class="x y z"`). Původně aplikace všechny uvedené hodnoty spojila dohromady (`xyz`), což bylo zapříčiněno funkcí `replace()` v kombinaci s regulárními výrazy. Chybu bylo třeba opravit, jelikož Reveal.js může vyžadovat umístění více hodnot (například při změně animace snímku `data-transition="fade-in slide-out"`). V aplikaci došlo k malé úpravě procesu zpracování vkládaných atributů, čímž byla chyba odstraněna.

## 6.3 Uživatelské testování

Na závěr si práci s aplikací vyzkoušelo pět osob, jejichž zkušenosti s HTML a CSS se lišily (všichni však měli základní znalosti). Těm byl sdělen základní účel aplikace a poskytnuta dokumentace frameworku Reveal.js, která je doplněna o mnoho názorných ukázek.

Uživatelé měli za úkol vytvořit krátkou prezentaci (nejméně pět snímků) na libovolné téma obsahující alespoň tři obrázky (jeden z nich použít jako pozadí snímku), uzpůsobit vzhled jednoho snímku pomocí vlastních kaskádových stylů, prohodit mezi sebou snímek č. 1 a 3 a dle svého uvážení změnit šablonu a nastavení prezentace. Zkoumala se hlavně rychlost, s jakou jsou uživatelé schopni prezentaci vytvořit. V případě větší překážky měli uživatelé možnost se dotázat na správný postup.

Z tohoto testování vzešlo několik zajímavých podnětů. Uživatelé zmiňovali, že pokud by jim nebyl předem sdělen účel aplikace, tak jej nedokáží odhadnout. Problém byl odstraněn přidáním popisu na úvodní stranu. Pokud uživatel zvolil jako cloudové úložiště Google Drive, tak nedokázal překonat varovnou hlášku nedoporučující používání aplikace. Ta se zobrazuje z toho důvodu, že aplikace momentálně není společností Google ověřena. Často se uživatelé snažili nahrát více mediálních souborů najednou, což však nyní není umožněno.

# Kapitola 7

# Závěr

Cílem práce bylo provést analýzu cloudových úložišť, jejich aplikačních rozhraní a prozkoumat možnosti integrace do webové aplikace. Nejdříve ze všeho bylo třeba se seznámit s dostupnými technologiemi, které umožňují tvorbu moderních webových aplikací. Následný průzkum aplikačních rozhraní cloudových úložišť a již existujících aplikací pro tvorbu prezentací položil základy pro návrh vlastního řešení.

Nastudované informace byly zužitkovány při zpracování webové aplikace, která uživatelům umožňuje vytvářet poutavé prezentace uchovávané ve vzdálených úložištích. Ty jsou k dispozici celkem čtyři – Google Drive, Dropbox, GitHub a Gitlab. Aplikace byla implementována za pomoci Vue.js a rozšířena o několik npm balíčků. Za nejdůležitější z nich lze označit Reveal.js, který dokáže dát jednoduchému HTML kódu podobu dynamické prezentace.

Aplikace poskytuje prostředky pro vytvoření zajímavé prezentace, která může být obohacena nejrůznějšími animacemi. Díky tomu mají její uživatelé příležitost snáze zaujmout pozornost svých posluchačů, což zejména v dnešní době vzdálené komunikace nemusí být vždy úplně jednoduchý úkol.

V budoucnu by aplikace mohla uživatelům nabídnout přidávání pluginů třetích stran do prezentace, tvorbu vlastních šablon či používání RevealNotes. Z uživatelského testování vzešel podnět k umožnění nahrávání více souborů najednou, čímž by se hlavně na začátku urychlila tvorba prezentace.

Z osobního hlediska vidím největší přínos práce v rozvoji vlastních dovedností. Nikdy dříve jsem neměl možnost používat Vue.js, respektive žádný javascriptový rámec. Díky práci se mi podařilo do této oblasti nahlédnout a naučit se základním principům.

# Literatura


[1] Andersen, B. *Vue Instance Lifecycle & Hooks* [online]. Coding Explained, 27. dubna 2017. Revidováno 11. 6. 2017 [cit. 2021-04-12]. Dostupné z: https://codingexplained.com/coding/front-end/vue-js/vue-instance-lifecycle-hooks.

[2] Barášek, J. *Komponenty ve Vue.js - Jak na to* [online]. Kladno: Jan Barášek, listopad 2019. Revidováno 7. 8. 2020 [cit. 2021-04-12]. Dostupné z: https://vue.baraja.cz/komponenty.

[3] Barášek, J. *Úvod do frameworku Vue.js* [online]. Kladno: Jan Barášek, listopad 2019. Revidováno 7. 8. 2020 [cit. 2021-04-12]. Dostupné z: https://vue.baraja.cz/uvod-do-vue.

[4] *Browser Market Share Worldwide* [online]. Dublin: StatCounter, březen 2021 [cit. 2021-04-06]. Dostupné z: https://gs.statcounter.com/browser-market-share.

[5] *Co je cloud – definice* [online]. Microsoft Azure, 2018 [cit. 2021-04-08]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/what-is-the-cloud.

[6] *Co je cloudové úložiště a jak ho využít* [online]. Microsoft Azure, 2020 [cit. 2021-04-08]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/what-is-cloud-storage/.

[7] *Co je webový prohlížeč?* [online]. Mozilla, 2020 [cit. 2021-04-06]. Dostupné z: https://www.mozilla.org/cs/firefox/browsers/what-is-a-browser.

[8] Connor, T. *Virtuální DOM* [online]. Tomáš Connor, 29. června 2019 [cit. 2021-04-12]. Dostupné z: https://tomasconnor.wordpress.com/2019/06/29/virtualni-dom/.

[9] Foltýn, M. *Databáze snů* [online]. Olomouc, 2018. [cit. 2021-04-12]. Bakalářská práce. Univerzita Palackého v Olomouci, Přírodovědecká fakulta. Dostupné z: https://library.upol.cz/arl-upol/cs/csg/?repo=upolrepo&key=6428218389.

[10] Hanák, D. *Stopařův průvodce REST API* [online]. Praha: David Čápka, 16. listopadu 2013 [cit. 2021-04-08]. ISSN 2464-6326. Dostupné z: https://www.itnetwork.cz/programovani/nezarazene/stoparuv-pruvodce-rest-api.

[11] Hassman, M. *Proč používat javascriptový framework?* [online]. Praha: Zdroják, 30. září 2008 [cit. 2021-04-08]. Dostupné z: https://zdrojak.cz/clanky/proc-javascriptovy-framework/.

[12] Hattab, H. E. *The HTML presentation framework* [online]. 2013 [cit. 2021-04-13]. Dostupné z: https://revealjs.com/.

[13] Hau, T. L. *Reactivity in Web Frameworks (Part 1)* [online]. Tan Li Hau, 2020 [cit. 2021-04-12]. Dostupné z: https://lihautan.com/reactivity-in-web-frameworks-the-when.

[14] Herout, T. *Dynamické webové stránky* [online]. HelpMark, 6. ledna 2012 [cit. 2021-04-06]. Dostupné z: https://www.helpmark.cz/slovnikpojmu/32-dynamicke-webove-stranky.

[15] Heřmánek, T. *Co je to PHP a k čemu mi bude dobré?* [online]. Praha: CoreIT, 24. března 2013. Revidováno 22. 1. 2015 [cit. 2021-04-07]. Dostupné z: https://www.coreit.cz/php/.

[16] *HTML History* [online]. W3cschool.in, 2020 [cit. 2021-04-07]. Dostupné z: https://www.w3schools.in/html-tutorial/history/.

[17] Koďousková, B. *Co je to API a jaké jsou možnosti jeho využití?* [online]. Praha: Rascasone, 13. února 2020. Revidováno 13. 4. 2021 [cit. 2021-04-08]. Dostupné z: https://www.rascasone.com/cs/blog/javascript-frameworky-spa-jednostrankove-web-aplikace.

[18] Koďousková, B. *Vue JS: výhody, nevýhody a možnosti využití* [online]. Praha: Rascasone, 5. února 2020. Revidováno 7. 1. 2021 [cit. 2021-04-12]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-framework-vuejs.

[19] Koďousková, B. *Vývoj webových aplikací s ReactJS, nebo VueJS?* [online]. Praha: Rascasone, 11. srpna 2020. Revidováno 13. 4. 2021 [cit. 2021-04-12]. Dostupné z: https://www.rascasone.com/cs/blog/vyvoj-webovych-aplikaci-reactjs-vuejs.

[20] Kuba, M. *OAuth 2* [online]. Brno: Ústav výpočetní techniky Masarykovy univerzity, 2018 [cit. 2021-04-08]. Dostupné z: https://is.muni.cz/el/1433/jaro2018/PA160/um/PA160-OAuth-2018.pdf.

[21] Lavičková, P. *Webová aplikace (Web Application)* [online]. Wilmington: ManagementMania.com, 10. prosince 2012. Revidováno 18. 10. 2018 [cit. 2021-04-06]. Dostupné z: https://managementmania.com/cs/webova-aplikace-web-application.

[22] Michálek, M. *NPM: Průvodce začátky a základními příkazy* [online]. Praha: Vzhůru dolů, 19. listopadu 2018 [cit. 2021-04-13]. Dostupné z: https://www.vzhurudolu.cz/prirucka/npm.

[23] Microsoft. *Začínáme s Vue CLI* [online]. Leden 2021, revidováno 25. 4. 2021 [cit. 2021-04-12]. Dostupné z: https://docs.microsoft.com/cs-cz/learn/modules/vue-cli-components/2-vue-cli.

[24] NPM. *About npm* [online]. 2019 [cit. 2021-04-13]. Dostupné z: https://docs.npmjs.com/about-npm.

[25] Přispěvatelé. *Document Object Model* [online]. Wikipedie: Otevřená encyklopedie, 3. ledna 2006. Revidováno 22. 5. 2020 [cit. 2021-04-08]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Document_Object_Model&oldid=18537143.

[26] Shinde, V. *10 BEST Free Cloud Storage Providers (Online Storage 2021)* [online]. SoftwareTestingHelp, 8. února 2019. Revidováno 2. 4. 2021 [cit. 2021-04-08]. Dostupné z: https://www.softwaretestinghelp.com/cloud-storage-providers/.

[27] Štráfelda, J. *Co je HTML* [online]. Velvary: Strafelda.cz, 10. března 2021 [cit. 2021-04-07]. Dostupné z: https://www.strafelda.cz/html.

[28] Štráfelda, J. *Co je JavaScript* [online]. Velvary: Strafelda.cz, 10. března 2021 [cit. 2021-04-07]. Dostupné z: https://www.strafelda.cz/javascript.

[29] *The authorization code grant flow* [online]. eBay Inc., 2019 [cit. 2021-04-08]. Dostupné z: https://developer.ebay.com/api-docs/static/oauth-authorization-code-grant.html.

[30] Thiessen, M. *Props Versus Data in Vue: The Subtle Differences You Need to Know* [online]. Michael Thiessen, říjen 2018 [cit. 2021-04-12]. Dostupné z: https://michaelnthiessen.com/vue-props-vs-data.

[31] Uličný, V. *Javascript frameworky: v čem programovat SPA?* [online]. Praha: Rascasone, 6. ledna 2020. Revidováno 13. 4. 2021 [cit. 2021-04-08]. Dostupné z: https://www.rascasone.com/cs/blog/javascript-frameworky-spa-jednostrankove-web-aplikace.

[32] Větrovská, P. *Úvod do CSS* [online]. WebTvorba, 2004 [cit. 2021-04-07]. Dostupné z: http://www.webtvorba.cz/css/uvod-do-css.html.

[33] Vue.js. *What is Vuex?* [online]. Vue.js, 2015 [cit. 2021-04-12]. Dostupné z: https://vuex.vuejs.org/#what-is-a-state-management-pattern.

[34] Vue.js. *Vue CLI* [online]. Vue.js, červenec 2019 [cit. 2021-04-12]. Dostupné z: https://cli.vuejs.org/guide.

[35] Vue.js. *Data Properties and Methods* [online]. Vue.js, říjen 2020 [cit. 2021-04-12]. Dostupné z: https://v3.vuejs.org/guide/data-methods.html.

[36] Vue.js. *Components Basics* [online]. Vue.js, 2021 [cit. 2021-04-12]. Dostupné z: https://vuejs.org/v2/guide/components.html.

# Příloha A

# Obsah přiloženého paměťového média

- **xstudn00.pdf** – technická zpráva
- **obsah.md** – obsah přiloženého CD
- **spusteni.md** – návod ke spuštění
- **/docs_src** – zdrojové soubory technické zprávy
- **/src** – zdrojové soubory aplikace

# Příloha B

# Spuštění aplikace

V počítači je třeba mít k dispozici node.js a npm. Získat je lze na níže uvedené adrese[1]. Při tvorbě aplikace byla aktuální verze nástrojů v14.16.0, respektive v6.14.11. Pro spuštění samotné aplikace je třeba provést následující kroky.

1. Zkopírovat obsah složky `src` do jiné složky v PC.
2. Přejít do této složky v terminálu.
3. V terminálu spustit příkaz `npm install`.
4. Ve složce `node_modules` odstranit složky `gdrive-utils`, `reveal.js`, `vue-file-agent` a `vodal`.
5. Obsah složky `node_modules_replace` zkopírovat do složky `node_modules` (kromě souboru `readme.md`).
6. V terminálu spustit příkaz `npm run serve`.

Po dokončení těchto kroků dojde ke spuštění vývojové verze aplikace. Ve výstupu příkazu `npm run serve` se nachází URL adresa, kde je aplikace dostupná.

[1] Stažení node.js a npm – https://nodejs.org/en/download/